Theses and Dissertations   1. Thesis and Dissertation Collection, all items

1994

Integrating computers into calculus instruction

Christensen, Jon L.; Pierson, Brian E.

Monterey, California. Naval Postgraduate School

http://hdl.handle.net/10945/30883

NAVAL POSTGRADUATE SCHOOL
Monterey, California

AD-A280 996

DTIC
ELECTE
S JUL 06 1994 D
F

THESIS

INTEGRATING COMPUTERS INTO
CALCULUS INSTRUCTION

by

Jon L. Christensen
and
Brian E. Pierson

March, 1994

Thesis Co-Advisors: Maurice D. Weir
                                       Carlos F. Borges

Approved for public release; distribution is unlimited.

DTIC QUALITY INSPECTED 3

94-20396

94 7 5 117

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE March 1994	3. REPORT TYPE AND DATES COVERED Master's Thesis	
4. TITLE AND SUBTITLE INTEGRATING COMPUTERS INTO CALCULUS INSTRUCTION			5. FUNDING NUMBERS
6. AUTHOR(S) CHRISTENSEN, Jon L. and PIERSON, Brian E.			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey CA 93943-5000			8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)			10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.	12b. DISTRIBUTION CODE A

13. ABSTRACT (maximum 200 words)

   Visualization is key in helping a student understand the fundamentals of Calculus. The new generation of computer literate students, raised in a video-based environment, will expect more than the traditional chalkboard methods in assisting them in this visualization. By integrating computers into the classroom and developing software to assist in mathematics instruction, we can enhance student comprehension of, and ability to apply, mathematics in solving real world problems of interest to the military.

   As evidenced by the success of both the Apple Macintosh and Windows software, mouse driven, graphical user interfaces (GUI's) represent a powerful and frequently-used tool in the computing environment. GUI's improve the visual capabilities in computing software, simplify program execution, and reduce the time required to become proficient with the software. When designed correctly, the GUI-based software can significantly improve the way in which people interact with computers. This thesis lays the framework and develops multi-platform GUI software modules needed for the instruction of Calculus.

14. SUBJECT TERMS Graphical User Interface, Multi-Platform, Portability, Color Look Up Table, Point and Click	15. NUMBER OF PAGES 110
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL

DTIC QUALITY INSPECTED 3

Integrating Computers into
Calculus Instruction
by
Jon L. Christensen
Captain, United States Army
B.S., United States Military Academy, 1984
and
Brian E. Pierson
Captain, United States Army
B.S., United States Military Academy, 1984

Submitted in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN APPLIED MATHEMATICS

from the

NAVAL POSTGRADUATE SCHOOL
March 1994

Author: _____
Jon L. Christensen

Author: _____
Brian E. Pierson

Approved by: _____
Maurice D. Weir, Co-Advisor

_____
Carlos F. Borges, Co-Advisor

_____
Richard Franke, Chairman
Department of Mathematics

ABSTRACT

Visualization is key in helping a student understand the fundamentals of Calculus. The new generation of computer literate students, raised in a video-based environment, will expect more than the traditional chalkboard methods in assisting them in this visualization. By integrating computers into the classroom and developing software to assist in mathematics instruction, we can enhance student comprehension of, and ability to apply, mathematics in solving real world problems of interest to the military.

As evidenced by the success of both the Apple Macintosh and Windows software, mouse driven, graphical user interfaces (GUI's) represent a powerful and frequently-used tool in the computing environment. GUI's improve the visual capabilities in computing software, simplify program execution, and reduce the time required to become proficient with the software. When designed correctly, the GUI-based software can significantly improve the way in which people interact with computers. This thesis lays the framework and develops multi-platform GUI software modules needed for the instruction of Calculus.

Accesion For		
NTIS CRA&I	☑	
DTIC TAB	☐	
Unannounced	☐	
Justification		
By		
Distribution /		
Availability Codes		
Dist	Avail and / or Special	
A-1		

iii

TABLE OF CONTENTS

ACRONYM LIST

CD-ROM . . . . Compact Disk - Read Only

CLUT . . . . .Color Look Up Table

CPU . . . . Central Processing Unit

DOS . . . . Disk Operating System

GUI . . . . Graphical User Interface

IBM . . . . International Business Machines

LCD . . . . Liquid Crystal Display

MB . . . . Megabyte

MHz . . . . Megahertz

PC . . . . Personal Computer

RAM . . . . .Random Access Memory

SVGA . . . . .Super Video Graphics Adapter

ACKNOWLEDGEMENT

I. INTRODUCTION

A. GENERAL

Calculus has traditionally been taught using illustrated textbooks and chalkboards. The arrival of the personal computer boom has opened new avenues for instruction that can greatly enhance the current methods. The purpose of this work is to investigate the integration of computers into calculus instruction. Our aim is to explore subjects and topics that are difficult to instruct using traditional teaching methods, along with those areas that may be improved by using the computer. The goal is not to produce a tutorial for the student, but rather to produce a multimedia tool that the professor may use in the instruction of calculus. Furthermore, the thesis offers sample modules to enhance the instruction of calculus, and presents classroom design and system needs essential for the successful integration of computer instruction.

B. OBJECTIVE

Humans are primarily visual learners. The more techniques and concepts that can be displayed visually, the better the potential for understanding on the part of the student. For example, merely explaining the meaning of slope as "the change in the x direction divided by the change in the y direction" is too abstract a discussion for many students. Current mathematics teaching philosophy introduces a simplified terminology such as "rise over run", and then an instructor usually draws some illustrations on the chalkboard to

1

demonstrate the idea. Generally, real understanding does not occur until the student sees the concept of slope drawn on the chalkboard. This capturing of visual representations of mathematical concepts is solely the burden of the instructor, and is frequently limited by his ability to draw. Even if the instructor refers to an illustration in the textbook, he or she must draw something on the chalkboard for purposes of reference during the classroom discussion. The integration of computers into the classroom environment can significantly lessen this burden.

Essential to our program design was the process of interviewing dozens of mathematics professors who have taught calculus for years. Collectively, their experiences pointed the way toward instructional areas that can be enhanced through the use of visualization techniques available on the computer. As a result of these interviews, several design requirements were identified. First, the design had to be easy to use. This requirement reduces training time and does not discourage inexperienced computer users. Next, each instructor expressed a strong desire to be able to enter their own equations and to alter the designs to conform to their own teaching style. Finally, the computer had to be a natural part of the instruction and not a distraction.

A review of the commercial software packages currently available revealed that, without a significant amount of preparation time on the part of the instructor, computer-aided instruction was nearly impossible. This identified a need for a system that can be readily and rapidly deployed by a professor for use in the classroom. The system should not replace the

2

professor's instruction, but should be a tool to aid in learning basic and abstract ideas. This thesis lays the groundwork for designing such a system, and introduces several example modules designed for use in the classroom.

The focus of this project is on the visualization of mathematical concepts. The ability to display quickly 3D graphics and animation that enhance conceptualization is critical to the design. The ability to display ideas interactively in the classroom is also essential if the instructional presentation is to be smooth and effective. The spotlight is on being able to better demonstrate mathematical concepts which are currently difficult, if not impossible, to present.

C. ISSUES

Several issues needed to be resolved before beginning a project of this magnitude. The first issue was determining the availability of computing platforms. To complete a project of this size in a reasonable amount of time, computers had to be dedicated from the Mathematics Department to support production. A second issue was procurement of the necessary software to create the modules. The final area of concern was time since a project of this scope takes several hundred hours to produce. Each of these issues had to be addressed in order for the project to move forward.

The first step was to solve the computer platform issue. Because the project was to be multi-platform based , two separate computer systems were needed. The fastest possible IBM based personal computer and Macintosh systems were selected because these systems

3

are most widely used. Specific descriptions of the systems involved and their development is addressed later. This portion of the development was probably the most time-consuming of the entire project because the computers had to be configured and installed onto the network to support production. This effort was time consuming, primarily because of a lack of documentation for the systems and lack of technical support.

Once the platforms were finalized, the focus turned to software. This is a process that can take several months because the procurement procedure can often be slow. The software packages called *Windowcraft* and *SuperCard* is physically in the domain of the Mathematics Department, so the production of future modules is now possible.

The last issue to be resolved was time. A project of this size requires over a year of initial preparation and execution time. Time invested included learning the basic operations of the computers, fundamental programming concepts, the operation of networking software, the building of windows interfaces, computer graphics handling techniques, and other computer related matters. The list is quite extensive, and a high degree of proficiency had to be reached in each area before moving forward with the project. Time management was the most critical issue to be resolved before starting production.

This project illustrates a rudimentary road map to create computer based instructional tools. It is these tools that the students will use to further their understanding and spark their desire to utilize and master the mathematical sciences.

II. TOPIC SELECTION

A. STARTING POINTS

Before starting the project, the areas of Calculus that warranted demonstration were chosen. The search began by writing down the index of <u>Calculus</u> [Ref. 1]. A review of the course syllabus of a core calculus course was also conducted. The combination of these sources gave a full range of topics that could be instructed during a normal calculus sequence. Once this list was compiled, areas that did not lend themselves to visualization techniques, or areas that did not follow previously described program tenets were eliminated. Topics that dealt with memorization, along with subjects that are generally presented non-graphically, were removed. For example, sections dealing with differentiation rules were removed since they are usually taught using analytical or operational methods. This first cut left a list of approximately eighty potential topics. The process of allocating the available time now had to be considered to tailor the scope of the project.

A total of eight months was available for the project. The first month was dedicated to the topic selection process. The second month was devoted to computer setup and learning basic programming skills. Production of the first module, along with project formulation meetings, consumed the remainder of the second month. The third month was dedicated to designing the control structure of the software. The remaining five months were committed to module production.

5

Initially, the plan was to prepare each of the eighty topics organized into twelve chapters. The Sine Function was chosen as the first module. The idea for this first module was simple. Using the general sine function formula

$$y = A \sin\left(\frac{2\pi}{B}x + C\right) + D$$

the module would plot various sine curves to illustrate the effects of changing the parameters A, B, C and D. This module took nearly three weeks to complete. Included in this time, however, was the overhead in learning the driving software and in becoming familiar with the programming environment. After careful analysis, it was determined that completion time for an average module would be slightly over one week. This estimate was validated throughout the project. This one week average completion time yielded a meter by which the number of modules possible for preparation could be estimated. The time line for the project allowed only four months for module creation, which necessitated cutting the initial list down to around sixteen topics.

Another constraint was the creation of the software that ties the modules together. After some experimentation with the design of the program architecture, it was evident that its construction would be time consuming. A month was allocated for building the program superstructure. This, in turn, left only four months to create the modules. With this additional constraint, the module goal was reduced to eleven. With the scope reduced to a realistic level, the selection of the topics for the modules began.

B. NARROWING THE AREA

The first module, the sine function, had already been created. This module was selected first because it would demonstrate and exercise the graphical capabilities of the computer. Additionally, the sine function is an easily understandable mathematical concept which would demonstrate the power of graphical computer methods for instructing calculus. Ten more modules remained to be selected. The focus for these modules was broken down into four major areas; concept, motion, animation, and instruction. These modules are listed below.

- Polar Plots
- Trapezoidal Rule
- Parametric Equations
- Projectile Motion
- Disk Method of Integration
- Undamped Spring
- Population Growth Model
- Cobweb Theory
- Simulation Model
- About...

The bulk of the modules would fall under the concept area. These modules were specifically chosen because they would help the student visualize high level mathematical principles. Each concept module generally begins with a standard mathematical equation, with options for parameter variation, as in the sine function module. It is critical in the learning process for students to understand the ramifications of altering the parameters, and to see the impact visually, such as illustrated in Figure 1. The modules that fall into this

concept theme were the sine function, polar plots, projectile motion, population growth
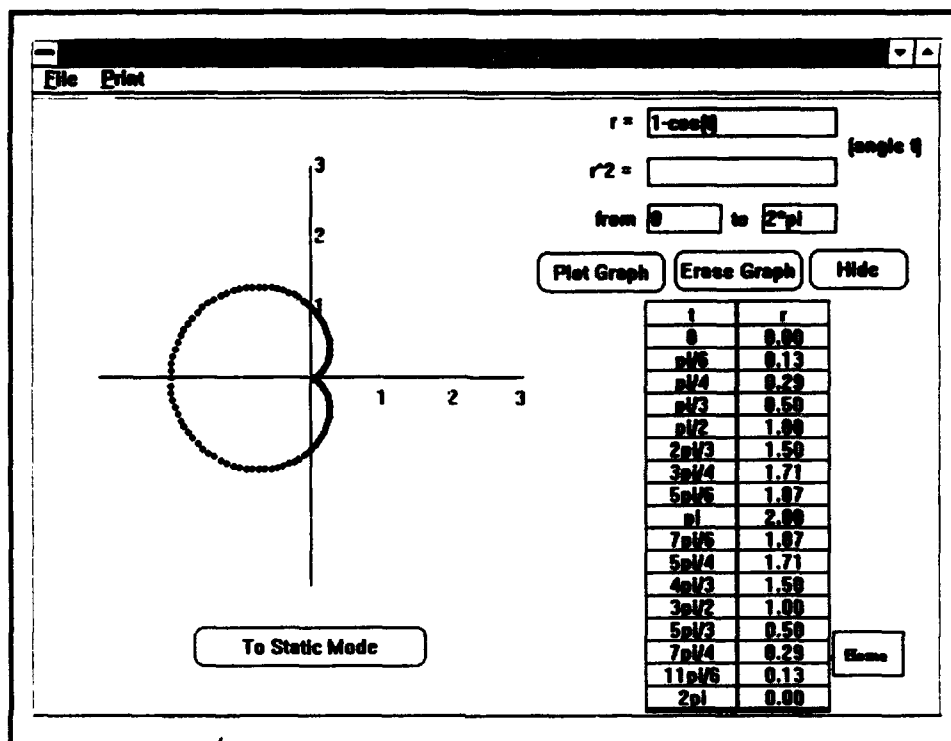
modeling, cobweb theory, and simulation modeling.

Figure 1 Polar Graphing
The cardioid 1-cos(t) is plotted dynamically on the interval 0 to 2π.

The next category of modules was motion based. One of these modules, the undamped

spring, shows the rectilinear motion of an undamped spring and its subsequent displacement

graph. The displaced cosine wave, as traced out by a harmonic oscillator, is exceedingly

difficult for the average student to visualize. With the aid of this module, the student can

readily see the mass bouncing up and down on the spring, and pictures its displacement over

time. Placing a spring animation on the screen together with a dynamically created plot of

its displacement, as shown in Figure 2, creates an effective visualization. Modules of this

8

type, which couple moving graphics with mathematical concepts, are highly successful in the instructional environment. The linking of real-world motion with the mathematical model representing its motion greatly enhances the student's comprehension.
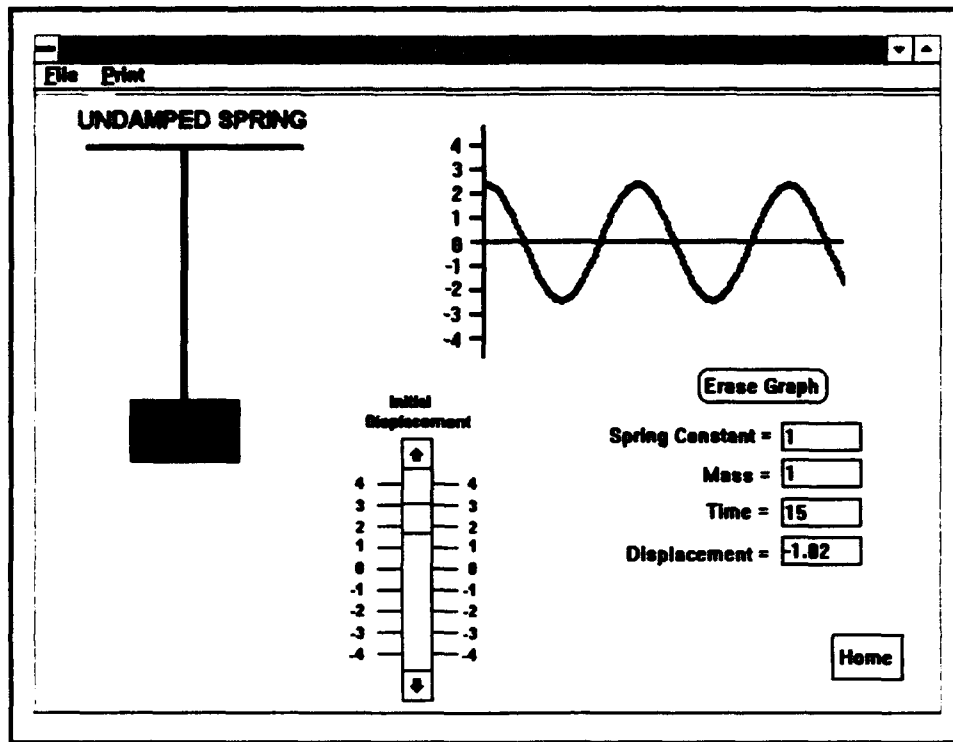

Figure 2 Undamped Spring
The spring moves up and down while the graph of the motion is simultaneously traced out.

The next group of modules was animation based. Describing the motion of objects is traditionally a challenge for instructors using conventional chalkboard techniques. The computer, however, opens the world of two and three-dimensional animation to make the task dynamic and simple. These animations demonstrate the appearance of an object moving through space. The two modules that display this idea present parametric equations and the

9

method of integration. The parametric equation module shows a cycloid traced by a point on a wheel rotated through time (see Figure 3).
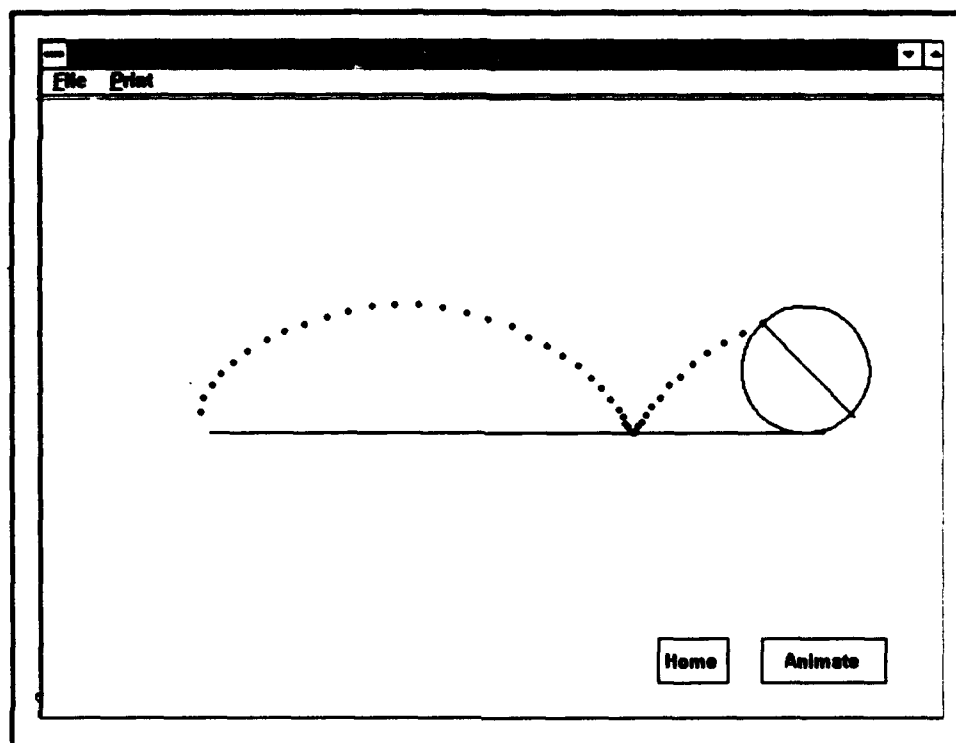

Figure 3 Wheel Cycloid
A cycloid is traced by a wheel rolling over a horizontal surface.

It is even more difficult to convey movement in three dimensions. While instructing integration using the disk method, the instructor has to describe the volume of a three-dimensional object generated by rotating a curve about an axis in space. The disk method module dynamically produces this rotation by assembling a collection of three dimensional bit-mapped graphics into a movie, which readily simulates the rotation of a curve about an axis in real time. Figure 4 shows the final result of this simulation technique. The concept of rotation is not presentable in class effectively with any other media.
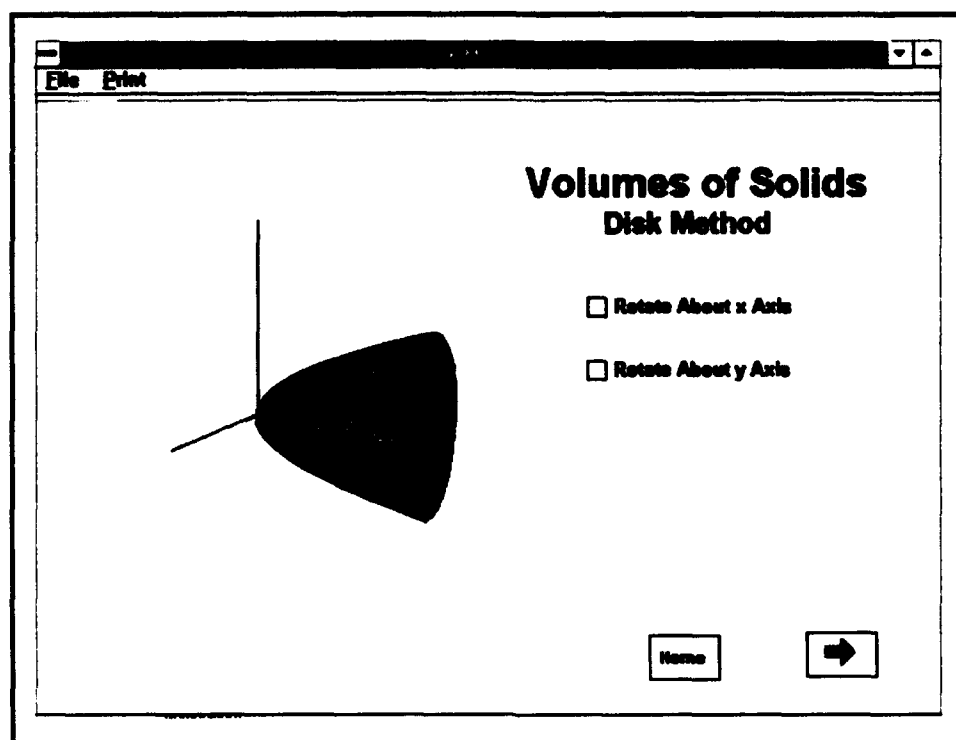
10

Figure 4 Disk Method
A solid of revolution is created by rotating a curve about the x-axis.

The final module category contains elements to enhance the instruction of a concept. These modules generally consist of a graphics library or slide show that can be used as either a primary or supplementary means of instruction. The trapezoidal rule is an example of this type of module. This module uses a slide show technique to illustrate how the area under a curve can be approximated by trapezoids (see Figure 5). These slide shows tend to be difficult to create because they require in-depth knowledge about the scripting software as well as about the underlying mathematical concept. All future modules will fall to some extent under the four categories described above. The base modules within these categories will serve as templates for future modules, thereby reducing their construction times. Once

an educator has a base knowledge of how a module is created in a given category, he or she

can refer back to that one as a basis for creating subsequent modules.
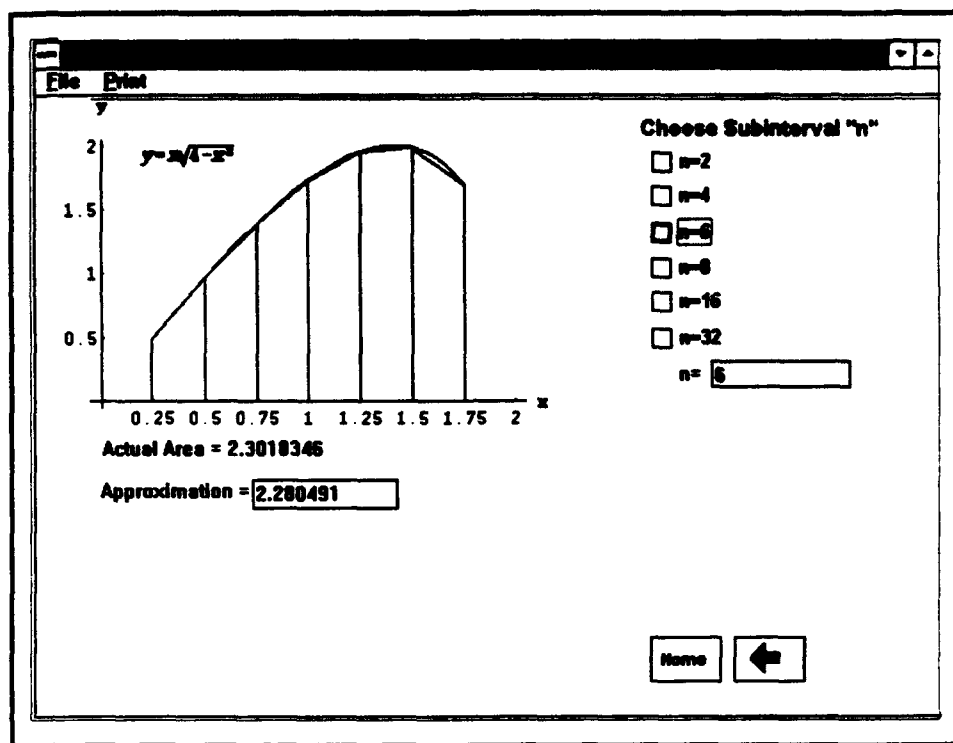

Figure 5 Trapezoidal Rule
The area under the curve is approximated by six trapezoids.

III. COMPUTING PLATFORMS

A. GENERAL

Since the project was to be computer based, it was necessary to select a suitable computer system for the software. From the reading of educational magazines directed at institutions of higher learning, it was evident that the preponderance of educational software was Macintosh based. However, the Department of Defense, to include the target audiences of West Point and the Naval Postgraduate School, traditionally uses IBM Personal Computers (PC) or clones. Targeting the PC users exclusively would result in the loss of potential Macintosh users. The goal became the production of software that both customers could rapidly implement. Therefore, the project was prepared for compatibility with both the Microsoft Windows environment and the standard Macintosh interface for the Macintosh. The design structure of the software, however, actually allows operation on all current platforms. The portability of the program gives it a definite advantage because it may be operated on any machine available to the user. No additional hardware or system upgrades are required to use the software.

B. PERSONAL COMPUTER

Availability became the driving force behind selecting a computer for the personal computer side of the design. The initial system made available for the project was a UNISYS 386, operating at a clock speed of 20 Megahertz. This system came standard with

four megabytes of random access memory, or RAM. This initial configuration, designed to operate in solely the DOS environment, proved to be unsuitable. The current operating standard is the windows operating system, which traditionally requires more RAM to operate efficiently. To program and use multitasking efficiently with the PC, a larger amount of memory was needed. The final RAM configuration for this machine before programming was eight megabytes. This computer also came standard with both 3½ inch and 5¼ floppy drives. Although initially these floppy drives were unremarkable, it was later found that the standard government machines have a low density 5¼ drive. Innumerable hours were spent determining why the drive could not read high density disks. While this problem appears trivial, it is exceedingly difficult for the first time computer user to make accurate hardware identification. To avoid this problem, it is best to have complete documentation on the machine before beginning any programming.

The storage capacity of the machine was also a critical concern. Graphics files are memory intensive and will fill rapidly a standard hard drive. The recommended hard drive size for storing graphics files is one gigabyte. While this may appear excessive, the estimate proved to be accurate throughout the creation of the animations and graphics required for the proper illustration of the module concepts. The UNISYS came standard with an eighty-megabyte hard drive which was too small to support the scope of the project. Because no other internal installation hardware was available, it became necessary to find an alternate solution to the storage problem. The network Novell server became the agent for alleviating

14

this problem. In addition to storage, the server had the added advantage of providing a variety of printers needed for the output of graphics and scripts. To use this server, the computer needed to be connected to the Naval Postgraduate School's network. To facilitate this process, it is highly recommended that the network procedure be standardized, well documented, and made available upon request to students in the future.

The final elements of the system were the monitor and peripherals. The monitor used is the color 640x480 SVGA. This monitor is the industry standard and is readily available. The peripherals included a three-button mouse and a keyboard. In summation, the initial computer configuration for the start of the project is shown in Table 1.

TABLE 1 INITIAL PC CONFIGURATION

Item	Description	Details
Central Processor	386	20 MHz
Math Co-Processor	386-20	Intel Co-Processor
Random Access Memory	8 MB	70 ns speed
Storage	80 MB	Seagate ST-251
Network Card	Intel	EXP16ODI
Video Card	Trident	SVGA 640x480

After working with this configuration for several months, it became evident that a faster processor was necessary for the timely completion of the project. A direct comparison was made between a 386/20 and a 486/33 machine. The resulting speed increase in module production on the 486 machine was remarkable. The significant increases were realized in

overall system operation speed and graphics handling. An analysis of systems revealed that the system listed in Table 1 is the minimum system configuration needed in producing the project. Using a slower machine would result in inordinate amounts of processing time and system waits in the production of output. The machine listed in Table 2 met or exceeded expectations in every category.

TABLE 2 FINAL PC CONFIGURATION

Item	Description	Details
Central Processor	486	66 MHz
Math Co-Processor	486	Integrated on CPU
Random Access Memory	16 MB	70 ns speed
Storage	340 MB	Quantum
Network Card	Intel	EXP16ODI
Video Card	Vortek 2 MB	SVGA 1200x900

It must be emphasized that these machines are not the machines required to run the presentation software. They are the machines recommended for use in designing and producing additional modules. The actual presentation software we have designed runs on all machines discussed in Section A. A single critical portion of any machine that will enhance operation is the 2MB Video Card. This device accelerates windows performance and graphic output.

16

C. MACINTOSH

The Macintosh system selected for programming was the Quadra 700 system. This platform offers speed combined with excellent built-in graphic handling capability. The Macintosh user environment is graphics based, so the architecture of the machine is designed to support the handling of graphics and window manipulation. The specific machine configuration is listed in Table 3. This system has all the capabilities necessary for module design, production, and usage.

TABLE 3 MACINTOSH CONFIGURATION

Item	Description	Details
Central Processor	68040	25 MHz
Math Co-Processor	Motorola	Built-in
Random Access Memory	20 MB	80 ns speed
Storage	2'0 MB	Apple-144
Video Card	Integrated	1.5 MB VRAM

IV. SOFTWARE

A. GENERAL

The project needed to take advantage of the latest advances in the computer industry. These advances, brought about through the standardization of personal computer interfaces, greatly simply programming concepts. One such advance allowed for creating the project modules without requiring a complete programming team. Specifically, the Windows operating system for the personal computer and the System 7 architecture for the Macintosh presented an environment where programs operate simply and quickly. The simplifying of programming techniques was also critical to the developmental process. One technique, called object-oriented programming, allows the programmer to create a single object and use it with slight modifications throughout the entire programming process. The user then needs only to activate that object in order to run the program. Previously, the user was often required to type numerous commands to carry out a particular task. Under object-oriented programming, designers can create software that allows the user to use a common-sense approach to running programs. An environment can be created that does not require sophisticated programming skills but only rudimentary knowledge of elementary computer operations to allow proper functioning of the program package. These advances in the information technology industry greatly empowered the project to move forward.

Selecting a programming utility for the project to take advantage of the technological advances was the most critical decision of the project. A language that was easily transferred

18

between the personal computer and Macintosh was desirable to avoid duplication of effort

inherent in building a multi-platform product. The product had to allow for creation of

objects that could be programmed to perform specific tasks. The software needed the ability

to draw dynamic graphs generated by codes internal to the program. The utility also had to

be based on the scripting techniques of programming. (Scripting, in this sense, is a method

by which programmers can write code in a form similar to written English.) The package

then, transparent to the programmer, transfers the code to machine language. As an example,

the following figure is a button, scripted for a specific task.

```
Erase Graph
```

This button is designed to erase any on screen plots, and to then reset any equations used to

create the plots. The program, or script, for this button is shown below.

```
on mouseUp
  get the rect of card button "graph plot"
  choose select tool
  drag from item 1 of it - 10, item 2 of it - 10 to item 3 of it + 10, item 4 of it + 10
  delete graphic
  put "" into card field "amp"
  put "" into card field "amplitude"
  choose browse tool
end mouseUp
```

The script is in a language called ClearTalk. An advanced computer user can easily read the script and determine its function. This script, upon activation of the mouse, selects the area of the screen in which graphics are plotted and also deletes the graphs. It will then reset areas of the screen reserved for equations to "empty strings" and then return control of the computer to the user (See Appendix B for a full description of ClearTalk and scripting). While this script may seem complicated initially, it is much easier to code than other higher-level programming languages. It is therefore ideal for programmers who lack a computer science background. ClearTalk provides an excellent vehicle for rapid program development that is both readable and readily understandable to an educator trying to produce a teaching tool. The minimal training time to learn the language coupled with its ease of use made scripting an essential capability for the software selected.

The final and most important characteristic was to have a stand alone product. The final product needed to operate independent of any other computational or graphics programs and in a classroom environment. This stand alone capability would significantly reduce the cost of operating the software, since the user would not be required to purchase other software to operate the program. It is also simpler to use than otherwise.

It next became necessary to find a software package that could meet or exceed these requirements. The starting point in the search for software was computer magazines and brochures. Initially this proved to be unsuccessful because these magazines are not primarily targeted toward programmers. However, it was learned that some professors had seen some

other graphics materials produced for physics using Claris's Hypercard. Using this information, a search was conducted for products produced in Hypercard. A mechanical engineering tool written at San Jose State University [Ref. 13] was located that appeared to have the functions necessary to make the project work. A software utility had now been identified that seemed to have the required functions and a family of programming products could now be investigated.

B. MACINTOSH

The search began for a utility for the Macintosh with Hypercard [Ref. 14]. This package, in a reduced form, is actually included with every Macintosh machine. The package, by itself, is capable of running all Hypercard products. Hypercard is a program that creates cards, similar to note cards, of any size and places them into collections called stacks. The stacks can then be manipulated to display any information contained on the cards. This appeared to be a very promising utility on which the project could be built. As a result, the full version of Hypercard, called the Developers Kit, was purchased at a price of just under one hundred dollars. This program appeared to meet all of the project requirements, so the first module was soon under way. The training time for learning the operation of the programming package was approximately one month. Most of this time was spent becoming familiar with the ClearTalk scripting vocabulary. Unfortunately, after working with the program for several months, a serious shortfall was identified. Hypercard was designed purely for black and white programming purposes. As such, it had no built-in capability

21

available for the importing of color graphics. In addition, its animation tools were very cumbersome. Because color greatly enhances the appearance of any instructional product, it became desirable to find a new utility that had color capability. The search for such a utility began and a scripting program called Aldus SuperCard [Ref. 7] was finally located. This program utility was touted to have full color capability and superior graphics and animation handlers. In addition, it was based on the ClearTalk programming language so after some further investigation, SuperCard was chosen as the program driver for the project on Macintosh. This scripting utility truly was everything it was supposed be! It handles color effortlessly and has superior on-line help as well as debugging routines. With SuperCard the creation of envisioned modules, within the given time frame, seemed to be a reasonable goal.

C. PERSONAL COMPUTER

Finding similar software for the personal computer proved to be a more formidable task. Because the Windows system is relatively new, not many programming utilities are yet available to the general public. The methods of locating software that were so successful for the Macintosh failed for the PC. Finally, two products that might meet the project needs became known. One such product was Microsoft's Visual Basic (which is the programmers choice for the creation of Windows based utilities). This product is designed for an experienced programmer and it requires detailed programming for all object-oriented functions. Another drawback is that Visual Basic uses a programming language which

22

differs significantly from ClearTalk (making it undesirable for this project). The second item was a little-known product called Windowcraft [Ref. 3]. This product uses virtually the same language and programming techniques as SuperCard and can directly translate Macintosh scripts. In addition, it has excellent graphic handling capabilities, can operate in a stand-alone mode with only the standard Windows software as a driver, and is relatively inexpensive. It, too, offers on-line help and excellent debugging features. As a result, Windowcraft was chosen as the software driver for the personal computer. The software met all the specified project requirements. Thus all programming utilities had been located that would allow for the creation of modules usable on both the personal computer and Macintosh machines.

V. FRAMEWORK DEVELOPMENT

A. CONCEPT

One of the critical design tenets for the program was ease of use. The program had to be user friendly in both operation and organization. A control structure was needed that would add a logical flow to the project. In addition, the structure had to be easy to manipulate. The combination of these needs led to the program's hierarchical or book design structure. A central controlling program runs all of the modules in the project. This control program, once initiated, gives the user a textbook feel (see Figure 6). First, the instructor selects a chapter for use. The controlling program then brings the available chapter modules into view. Next, the user chooses a specific module. This control program is analogous to a tree structure. The user must move vertically within the program, and horizontal movement is not allowed. This type of tree structure greatly simplifies the use of the program.
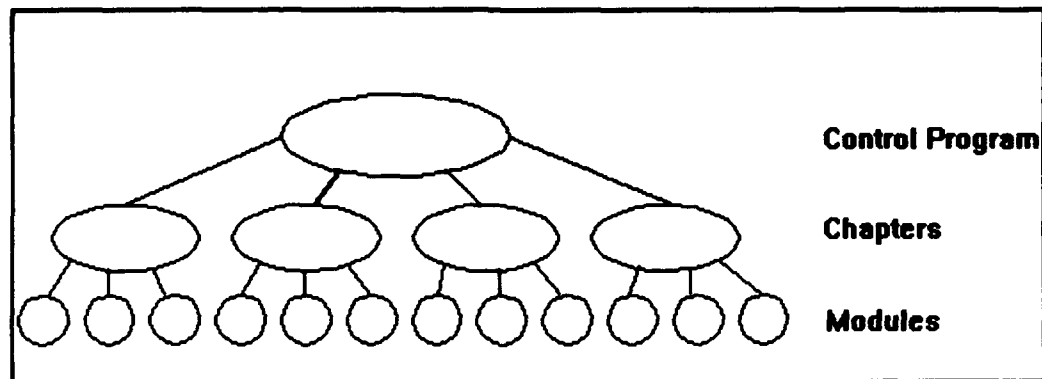
Figure 6 Program Hierarchy

The modules are organized into chapters, following the usual pattern of presentation in a standard calculus text. The chapters are then collected under the central program, giving the program its textbook structure. This type of design allows the user to find topics quickly and controls the flow of information in a logical pattern. The pattern allows for a program design structure known as point and click. Point and click is an operating design that allows the user to traverse the structure in a logical and easy to follow manner. The structure returns the user to chapter listings in the same way for each module. Similarly, each chapter's controls for moving up and down the structure are consistent throughout the entire design. Therefore, once the user becomes familiar with the control features he or she will only need to point and click the appropriate graphic to move from place to place throughout the tree.

B. PERSONAL COMPUTER WINDOW DESIGN

The personal computer design takes full advantage of the windows operating system. The controller program is summoned through an icon defined within the program. The user initiates the program by double clicking on the icon to start the program. At his point the program's logo, *CalcAid*, appears on the screen. The main *CalcAid* window initially displays a design screen along with standard "File" and "Print" menus located on a menu bar (Figure 7). The "File" menu includes options for quitting the program along with a subroutine entitled "About...". This "About..." function displays information about the program and its designers. The "Print" menu allows the user to print slides within the

program. By selecting the menu button, the user reaches the main menu, which is the heart of the controller program.
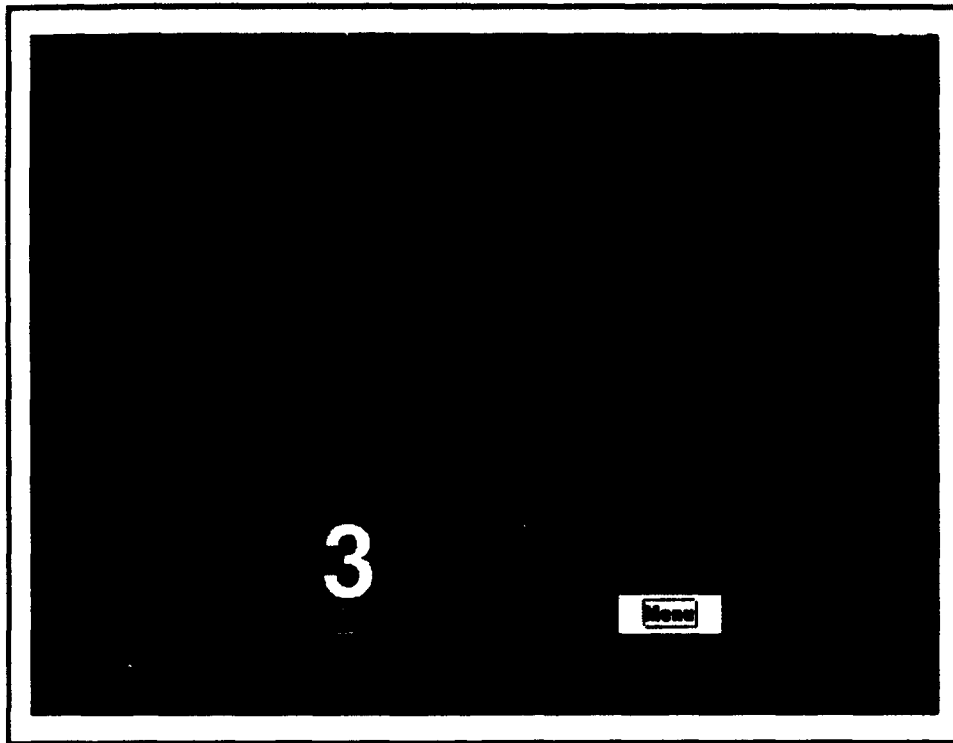
Figure 7 Opening Screen

The main menu gives the instructor the option to select a specific area of calculus for use. This chapter menu is shown in Figure 8. By double clicking on the subject area name, the user calls up the desired module. This structure speeds the use of the program and makes the design easy to follow. All screens display the menu bar that allows the user to terminate the program and instantly return to the windows environment.
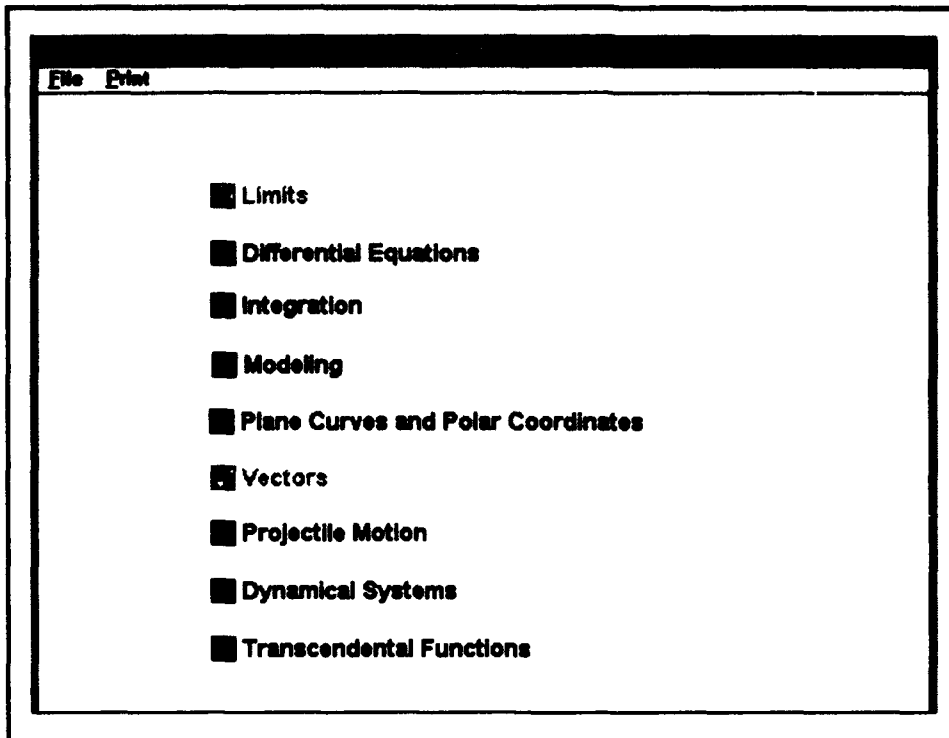
Figure 8 Chapter Menu

Once a chapter has been selected, the program displays a subject sub-menu. This sub-menu lists the modules available to the user within that chapter. The selection of a module from a sub-menu works on the same point and click principle as the chapter menu slide. By selecting an option from this screen, the instructor calls up a specific module for use. The controller program then opens a new window that will run and display the module selected. For example, by selecting the Trapezoidal Rule entry from the Integration sub-menu, the user would see a screen that looks similar to Figure 9. Further control measures are shown in this figure. These control features include the Home Button and the Left/Right Arrow Button. By selecting the Home Button, the user will return to the sub-menu previous to the given slide. The Left/Right Arrow Button either takes the user to the next slide, or returns the user

27

to a previous slide. These conventions are consistent throughout the program. Thus, the user must only possess rudimentary point and click skills to be able to navigate successfully throughout the program.
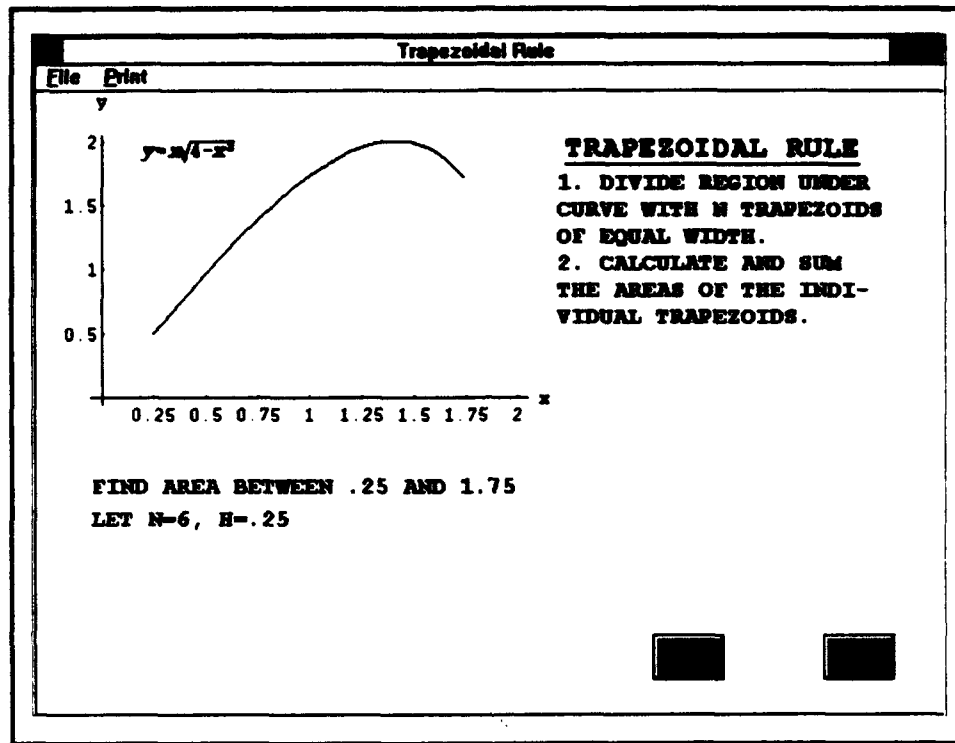
Figure 9 Trapezoidal Rule

C. MACINTOSH DESIGN

The operation of the program on the Macintosh is essentially the same as that of the personal computer. The operation of the control structure transparent to the user is the main difference. The Macintosh program consists of only one executable program. This program contains the controller program and all necessary sub-programs to operate the product. This structure conforms to the stack and card protocol habitually associated with the Macintosh.

28

VI. PROJECT DEVELOPMENT

A. MODULE DESIGN FACTORS

Before constructing modules for the program, several factors had to be considered. Interactivity, animation, and use of graphics are prime advantages of computer-based design. The project had to make maximum use of these benefits. The building of graphical user interfaces, or GUI's was also critical to the design. A module that is well laid out and easy to follow facilitates the program operation. The modules also had to be aesthetically pleasing, so physical layout and color selection became important design factors. While not all inclusive, this list of factors had a major influence on project preparation.

1. Interactivity

As stated above, the modules had to make full use of the interactive capabilities of the computer. The resulting module designs allow for changing equation parameters and graphically demonstrating the results of those changes. These graphic results give the student instant visual response to parameter changes. The Projectile Motion module illustrates this idea. The professor can begin by plotting the flight of a projectile with set parameters, such as shown in Figure 10. The professor and class may then speculate on the results of increasing the velocity by fifty units. With a minimum of effort, the instructor could then illustrate the results of the parameter change (see Figure 11). This ability to obtain direct results to different variable input greatly enhances the learning process and allows the professor and students to ask "what happens if..." questions. In addition, since

29
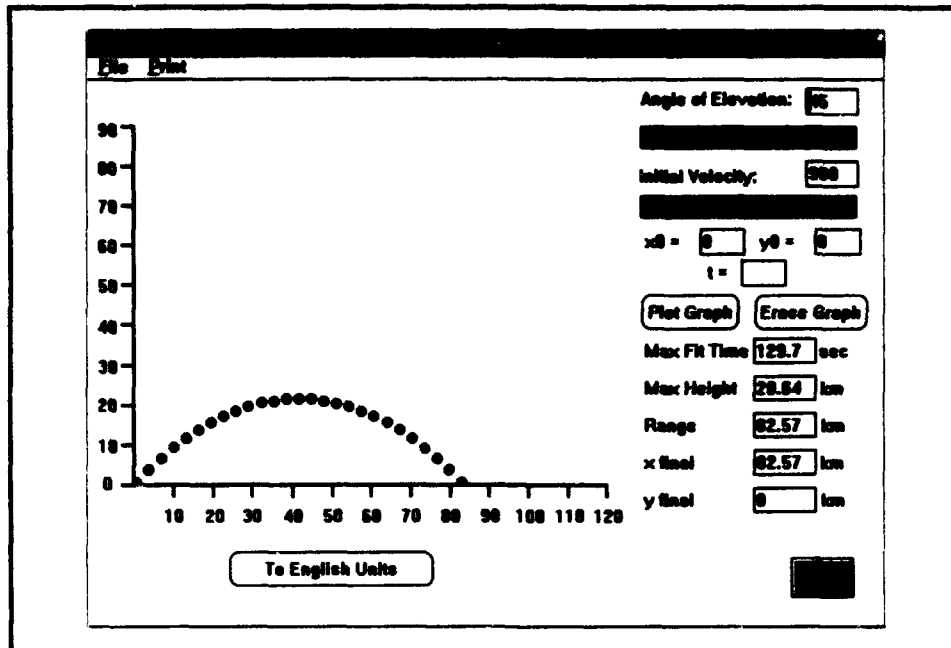
Figure 10 Projectile Motion 1
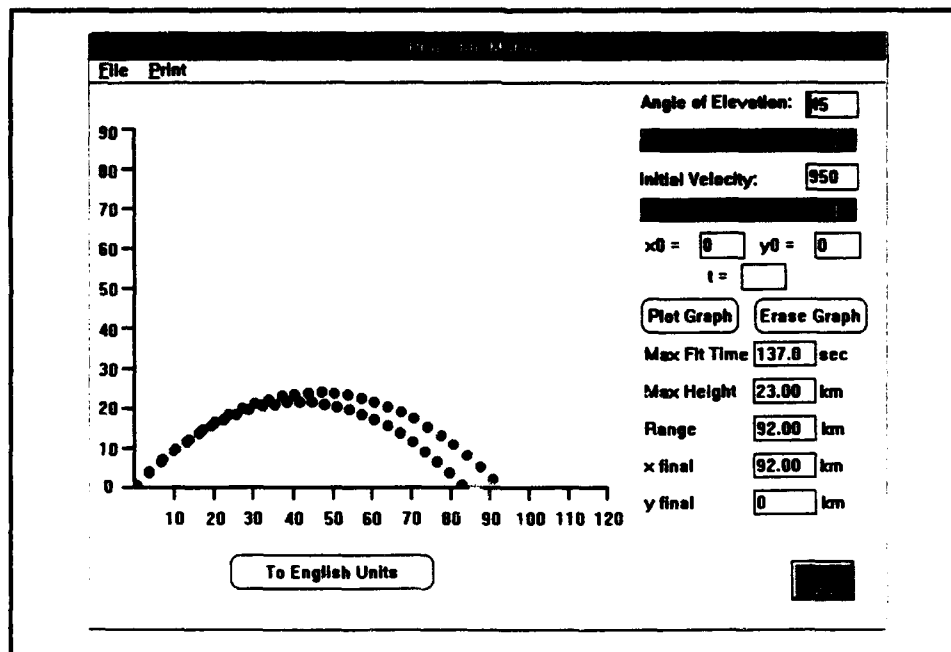
Figure 11 Projectile Motion 2

30

time and effort required to physically graph the equations is no longer necessary, the instructor can save valuable class time. This can only result in a better trained student.

2. Three-Dimensional Graphics

The ability to graph equations in three dimensions on a chalkboard is a challenge to all but the most artistically inclined mathematics professors. Furthermore, few students visualize well in three dimensions, and have difficulty forming a clear picture of what the mathematical equations mean geometrically. The computer can alleviate both problems. Taking advantage of the computer's graphic capability, several modules were constructed of pre-made bitmapped plots of equations in three dimensions. By bitmapping these plots, the instructor can get immediate response to input, and does not have to wait extended periods of time while the computer generates images. These plots accurately depict the characteristics and shapes of the figures to the students. The simplest case is a collection of bitmapped images that help students visualize cutting planes and conic sections. A more complex case involves the use of animation, and is demonstrated in the Disk Method Module. As pictured in Figure 12, one aspect of the module shows the results of rotating a curve about the x-axis. The computer's ability to graphically portray equations in three dimensions is one of the driving forces in integrating computers into the classroom. By taking advantage of this capability the computer has a superior means of conveying difficult concepts to the student.

31
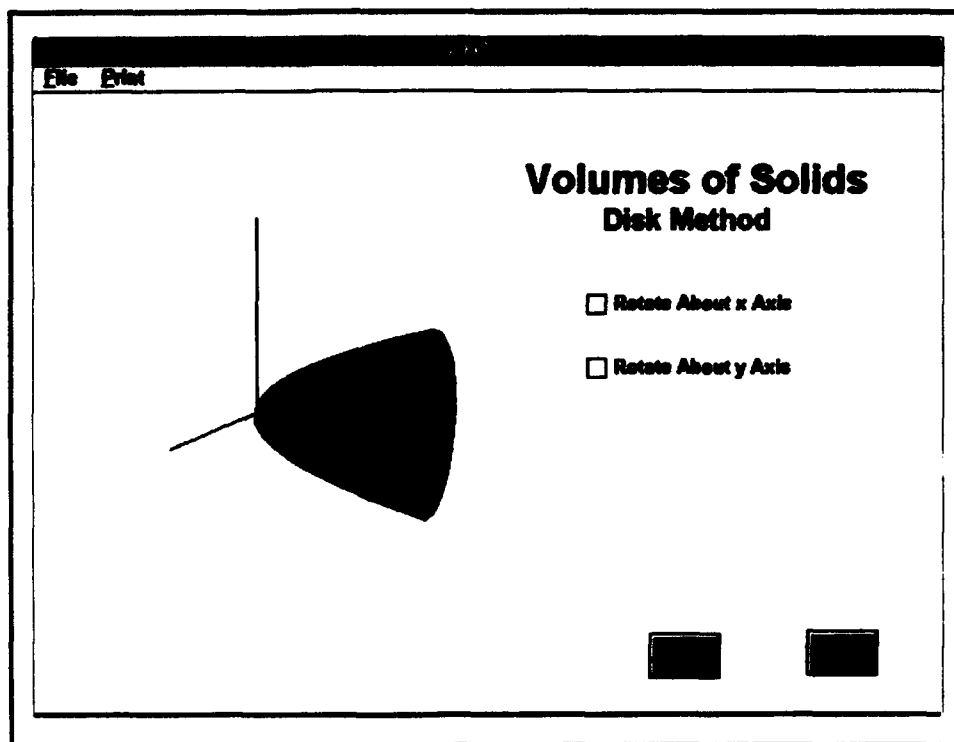
Figure 12 Disk Method

3. Animation

The computer offers an excellent means for adding motion to module presentations.
By creating a series of graphics and displaying them in rapid succession, like movie frames,
the illusion of motion is created. This technique is a powerful tool in presentation of
mathematical ideas. Figures 13 through 16 show the path of a rolling circle on a smooth
surface and illustrate this animation technique.   Under normal classroom situations an
instructor would have to trace out the circle's path on the chalkboard to demonstrate the
cycloid concept. This technique is both time consuming and cumbersome. With the
computer, however, the professor can simply activate the pre-made animation and the cycloid

path is drawn through an animation sequence. The students quickly see a visual image that helps them understand the concept.
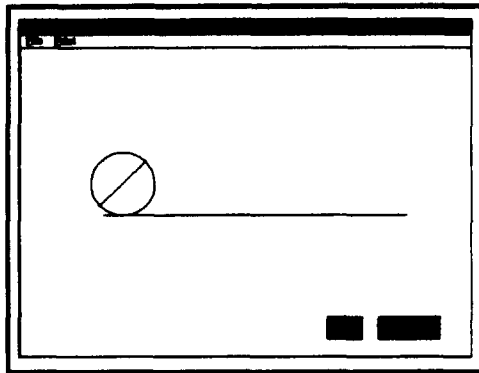
Figure 13 Cycloid 1
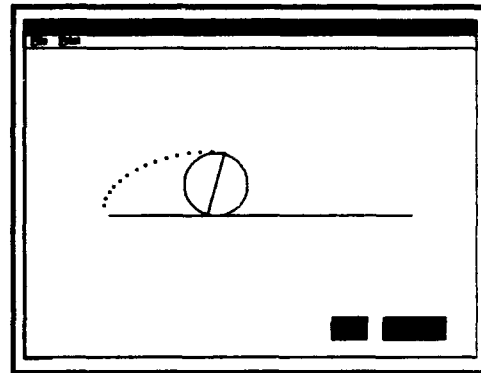
Figure 14 Cycloid 2
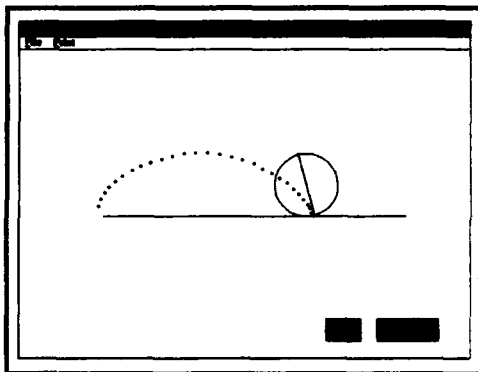
Figure 15 Cycloid 3
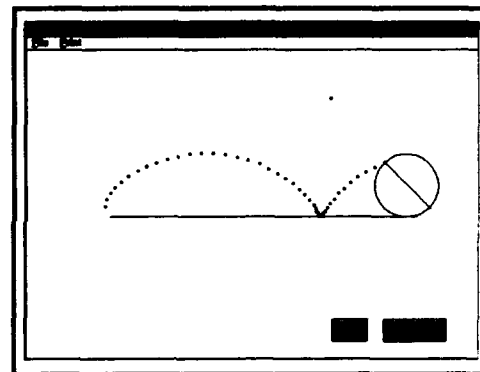
Figure 16 Cycloid 4

4. Interface Building

The careful design of the graphical user interface became a key component of module production.   The modules had to be easy to use and had to follow a convention easy to learn. An object's function had to be readily identifiable to the program user. In addition, the physical layout of the modules had to be visually appealing. By applying these design tenets to the project, an aesthetically pleasing and functional interface was created.

33

It was important to maintain a consistency of design throughout the project. Since the program adheres to a set structure, the user needs to learn only a few commands. The project's navigation tools and basic operating structure remain consistent through all modules. For example, a Home Button always takes the user back to an initial slide. Thus, by learning the conventions of one module, the user can effectively manipulate the entire package.

Individual modules were built following the logical design convention. Each module was designed in turn with the same point and click structure. Input fields and buttons were fashioned to draw upon the user's common sense and then placed on the screen in carefully planned locations. Each item on the screen connects an activity with an action. Interfaces were designed to eliminate the need for a user's manual. This, in turn, makes the product more likely to be used because a person can immediately begin running the software with minimum effort. This type of layout simplifies use and reduces the time spent learning to run the program.

The artistic aspect of design was not overlooked. The modules were built so that the layout and color selection were not egregious. Colors were carefully selected to complement the design of the module. Similar colors were chosen to present a module's structure, while contrasting colors were selected to stress concepts. Color was incorporated into the production of all graphics, so that specific aspects could be emphasized. Design considerations were not limited to color; the physical layout of the screen was also important.

Figures and objects were carefully placed on the screen to give a sense of proportion to the modules. A great deal of time was spent improving the appearance of the computer screens in the modules. A project of this type must be pleasing to the student's eye and also functional if it is to be effective and useful as an instructional tool.

B. TECHNICAL DESIGN FACTORS

There are currently a great number of central processing units, or CPUs, available in today's market. The program needed to be compatible with all of them which is a challenging design constraint. Each of the CPUs have different speed and graphic handling capabilities. More importantly, the computing standards of the processors often differ and they do not all operate on the same floating point arithmetic standard. In designing the project, it became essential to find consistency between the two platforms in all three of these areas. Although the three technical problems addressed here are not all inclusive, they do represent fairly well the areas dealt with in formulating the project.

1. Processing Speed

The processing speed of the computer was critical in furthering module design. The project was initially designed to function on the 386 and 68040 families of processors. After several design reviews, it became apparent that the project needed to be capable also of running on slower machines. Because many of the dynamic and animation modules were already at their processing limit, this posed a design problem. When these modules were run on slower machines, screen output was excessively slow and frequently resulted in

35

system crashes. Additional programming and other options were needed to correct these shortfalls. As a solution, static operation modes were created for several memory hungry dynamic modules. The static option consists of predrawn images that replace the dynamic plotting in given modules. These predrawn images were selected as representative of their entire group, and can be called up instantly by the user. As a result, processing speed was greatly reduced and the lower end machines no longer failed. Unfortunately, the interactive and animation capabilities of several modules are lost when dropping to the slower processors. But with the addition of static modes, the slower machines can still convey many of the module teaching points.

2. Floating Point Arithmetic

The computer performs mathematical operations using floating point arithmetic, (expressed more simply as just floating point). Several of the modules contained within the project possess codes that require numerical calculations in order to display graphic results. The inexperienced computer programmer often assumes that all computers operate in the same fashion, but in reality this is almost never the case.

Many people consider floating point arithmetic an esoteric subject. However the floating point problem is ubiquitous in computer systems, and quickly became a factor in the design of this project. The script language used in the development of this program has a floating point data type. The means by which the various microprocessors available on the market handle floating point differ greatly, and unfortunately do not adhere to the same

36

standards. This became painfully obvious when the software was transported across different machines or platforms and erroneous or unexpected numerical results often occurred.

The floating point problem was first encountered using the Trapezoidal Rule module. In this module, the computer performs thousands of numerical calculations to approximate the area under a curve by summing together many inscribed trapezoids approximating the curve. The module was originally created on the UNISYS system and was functioning correctly. When the module was transferred to the 486 system the different processor caused floating point errors to occur. The cause of the problem was traced to the way the machine performed floating point. This problem was quite unexpected to a novice programmer. An investigation yielded that the current generation of personal computers do not all adhere to the IEEE 754 standard [Ref. 9]. The problem was not insurmountable but its solution required the introduction of double precision number formats into the computer code. This, in turn, resulted in reduced calculation speed in some of the modules. The use of double precision is by no means to be considered a panacea for all floating point problems. It does however appear to cure a majority of the difficulties we encountered by moving from processor to processor. The need for the programmers to have a basic understanding of floating point arithmetic can not be over emphasized.

3. Color Displays

The selection of color schemes became a problem during the course of the project, especially for the personal computer. Typical video displays on the market show anywhere

37

from 16 standard colors to over 256 colors. In programming color selection, a number is assigned to each color choice. The specific color number is usually found in the given program's Color Look Up Table, or CLUT. Unfortunately, these CLUTs are not standardized for all Windows based programs. In addition, not all video drivers display colors in exactly the same way. As a result, colors schemes were often changed "behind the scenes" when transporting the program across platforms or to other machines. Color mapping was difficult to predict or standardize, and some machines even rendered the dules unreadable. To further complicate matters, the first version of the Windowcraft driving software had built-in internal problems when dealing with 256 color design. To simplify matters, and to eliminate the color mapping problems, all personal computer modules were designed using only the standard 16 color mapping. The next version of Windowcraft will have better color handling capabilities and future modules can be developed using the full 256 color spectrum.

VII. MODULE DESCRIPTIONS

A. POLAR PLOTS

The Polar Plots Module graphs polar equations in both the r= and the r^2= formats. This module was designed to assist the student in envisioning plotting in polar coordinates. In the dynamic mode, this module will overplot a number of manually entered polar functions. The first four functions are plotted in different colors to assist in visualization. At the user's discretion, a table is produced showing the value of the polar equation at standard angle intervals. This table helps the student understand function symmetry and how polar functions are actually plotted. Scaling is automatic and adjusts to the input equation.

The static mode consists of twelve predrawn options, including circles, cardioids, lemniscates, limiçons, and four-leaved roses. In this mode the user may rapidly choose and overlay the predrawn equations in order to show curve sizes, shapes, and intersections. Since the curves are predrawn, speed is rapidly increased over that of the dynamic mode. This static mode is useful on slow-processing machines.

B. SINE FUNCTION

The Sine Function Module plots the standard sine function in both static and dynamic modes. This module was designed to facilitate the understanding of the general sine curve by enabling the user to plot the general sine function and adjust parameters. In the dynamic

mode, the user may vary the parameters as indicated by the general sine function equation of the form

$$f(x) = A \sin [B(x-C)] + D$$

where A is the amplitude, B is the number of periods desired in a 2π interval, C is the phase shift, and D is the vertical translation. Separate functions are plotted simultaneously, with the first five being of different color to facilitate visualization. In order to simplify scaling, the user is limited to an amplitude between ± 5; phase shifts of 0, $\pi/2$, π, $\pi/3$, or 2π; number of periods in 2π between 0 and 8, and translations between ± 2.

The static mode was designed for use on lower end machines. In this mode, the user may choose to display any of twelve different predefined sine functions. All functions may be displayed simultaneously, and all are drawn using different colors. Since the functions are predrawn, the demonstration speed is increased significantly over the that of the dynamic mode.

C. TRAPEZOIDAL RULE

The Trapezoidal Rule module gives an introduction to numerical integration methods using the Trapezoidal Rule. The module starts with an example of the graph of the function $y = x\sqrt{4-x}$, and steps through the process of finding the area underneath this curve between .25 and 1.75. The demonstration uses 6 as the number of subintervals with length h of .25. Sequential slides show individual trapezoids drawn underneath the curve and the calculations needed to determine the approximate area. The trapezoidal area is then shown

40

together with the actual area, and the trapezoidal figure clearly shows where the area differences occur. The final slide allows the user to visually portray the same function with a choice of six different prebuilt subintervals, or to numerically calculate the area under the function with any subinterval. This slide clearly demonstrates the error which occurs using small subintervals, and how the actual area and trapezoidal area become comparable at a larger subinterval of 64.

D. PARAMETRIC EQUATIONS

The Parametric Equations module demonstrates parametric equation concepts through the use of cycloids. In the wheel cycloid example, a wheel of constant radius rolls along a horizontal surface without slipping. In the animation sequence, the path traced by a point on the wheel's surface is plotted in sequence. At the end of the animation, the curve of the form $P(x,y)=(at+a\cos\theta, a+a\sin\theta)$ is clearly shown on the screen. In the square cycloid sequence, a square travels along an undulating surface without slipping. The path traced by a point on the vertex of square is then plotted in sequence. At the end of the animation, a curve similar to that of the wheel cycloid is shown.

E. PROJECTILE MOTION

The Projectile Motion module illustrates the concept of vector and parametric equations for ideal projectile motion. The calculations in this module are based solely on the ideal motion, assuming only gravity affects the flight of the projectile, factors such as drag and

frictional force are not taken into account. The equations of motion are

$$x = x_0 + (v_0 \cos \alpha)t, \qquad y = y_0 + (v_0 \sin \alpha)t - \tfrac{1}{2}gt^2$$

with (x_0, y_0) as the initial point, v_0 is the initial velocity, g as the gravitational force, and α the firing angle measured in degrees from the horizontal. This module allows the user to compare trajectories by changing the parameters of the equations: the initial velocity, angle of elevation, firing location, or time. Output includes graphic representation of the projectile motion, maximum flight time, maximum height, maximum range, and final height and range. The user may run the module in either a metric mode or an English units mode with output in the correct units. This module may be used to solve problems involving only one unknown parameter, and is useful for exploring problems consisting of more than one unknown parameter through iterative methods.

F. DISK METHOD

The Disk Method module is an outline of finding the volume of a solid of revolution using the disk technique. First, a curve is shown in the xyz-plane. The user is then prompted to rotate the curve about the x or the y-axis. The solid of revolution is generated about the proper axis using an animation sequence, which helps the student visualize the volume generating concept. The next slide shows a curve in the xyz-plane with rectangles drawn underneath to represent the generating region. At the user's prompt, the rectangles are incrementally revolved around the x-axis, forming the solid disks whose volumes approximate the volume of the solid. The module then portrays the idea of the "kth" disk

42

volume, the concept of the disk volume sum, and the transformation of this Riemann sum into an integral expression. Finally, the module demonstrates the entire procedure by rotating the curve $y = \sqrt{2x - x^2}$ about the x-axis, generating a half sphere, and finally calculating this familiar volume using the disk method.

G. UNDAMPED SPRING

The Undamped Spring module plots interactively the motion of a mass on a vibrating spring with no damping and without an external force, following Hooke's law. The module includes an animated sequence of a mass on a spring moving up and down in a vertical plane, given some initial displacement. As the mass begins to move, the equation of motion is plotted simultaneously on the screen. The module gives the student a clear picture of simple harmonic motion.

The equation of motion is modeled by the second-order differential equation

$$mu'' + ku = 0$$

with a constant mass m, a spring constant k, and a displacement from the equilibrium point at time t of $u(t)$. The general solution of this equation becomes

$$u = A \cos \sqrt{\frac{k}{m}} t + B \sin \sqrt{\frac{k}{m}} t$$

with the constants A and B determined by the initial conditions. The user provides one initial condition by entering an initial displacement u(0). The second initial condition is provided by assuming that the initial velocity u'(0) is zero. The solution, when plotted,

43

is a displaced cosine wave representing the simple harmonic motion of the mass. Since no external force and no damping is present, the curve maintains a constant amplitude and does not diminish with time.

The user may vary a number of parameters to illustrate changes in the equation of motion. The mass, spring constant, initial displacement (limited in range to ±4), and time may all be entered separately. Output includes the plot of the solution of the equation of motion and the displacement at some final time t. Multiple equations may be overplotted in different colors to show the effects of the parameter changes.

H. ABOUT...

The About... Module is an interactive program designed to tell the user about the software. An "about" sequence is common in most Windows driven software. Although this is not technically a mathematically based module, it does demonstrate the potential for animation in future programming. When the about function is activated, the primary screen of the project disappears. In its place a smaller sized window pops on the screen and a paint field with the project logo enters the window from the left. The paint field moves across the screen horizontally until it stops in the center of the window. At this time, two other paint fields drop in from the top of the window. The first paint field contains the name of the authors and the second window contains information about the project. These paint fields pass over and under the logo field, giving the illusion of layering, while traveling vertically down the window. Finally, a reset button enters from

44

to a fixed location on the screen. When the user clicks the reset button, the original project window reappears. These animation concepts and programming techniques can be used to illustrate advanced two and three-dimensional concepts in later modules.

I. POPULATION GROWTH MODEL

The Population Growth Model is designed to show the student the behavior of the logistic function as applied to population modeling. The logistic equation used is in the form $A(n+1)=(1+r)A(n) - bA^2(n)$, where n is the number of discrete units, r is the unrestricted growth rate, L is the carrying capacity of a population, and $b = r/L$. The user may alter the model by entering different parameter values for the growth rate, carrying capacity, number of periods, and initial population value. When the graph is plotted, discrete values for the population are plotted based upon the number of periods. From this graph the student can see readily if the population converges or diverges. An additional feature allows the user to show the population limiting line L along with its discrete plot.

J. COBWEB THEORY

The Cobweb Theory module is designed for two purposes. First, it uses the graphical technique called cobwebbing which enhances the student's understanding of dynamical systems and fixed points. Second, it helps the student understand the behavior of the logistic equation. The module uses the same logistic function form as in the Population Growth module described above.

45

In the Cobweb Theory module, the user begins by defining the parameters in the logistic equation by entering values for the growth rate, number of discrete units, and carrying capacity. Next the graph of the function as defined is plotted on the screen simultaneously with the graph of the function $y = x$. The user then enters an initial value for the population and the module then dynamically draws the cobweb for the dynamical system and returns an equilibrium value. With the aid of this module the student can see readily if the system is stable, unstable, or inconclusive.

K. SIMULATION MODEL

The Simulation Model was built in response to an actual problem assigned in a mathematical modeling course. It is included to demonstrate the potential for graphical programming in teaching modeling and simulation. Graphical changes which take place because of external input can occur visually in real time on the computer screen. This feature pays huge dividends in helping the student understand simulation methods.

This module models a distribution problem typically found in a Monte Carlo simulation. Specifically, students begin to line up to make course changes prior to starting an academic term. They must wait until a counselor becomes available to perform one of six different services. Some students may require only one service while others require two. The simulation assigns students to a service based upon a given historical distribution.

The user inputs the number of students taking part in the simulation. A random number generator then assigns each student to a service based upon the given distribution. As

46

students are assigned to a service, a bar chart is constructed on the screen based upon that assignment. A background bar chart remains in place which is based on the actual distribution. At the end of the simulation run, the user is able to compare immediately the simulation distribution with that of the target distribution by examining the two bar charts. Additionally, output is available showing the total and average number of students assigned to each service. This module is an excellent tool for showing students how a random number generator used in a Monte Carlo simulation actually builds a distribution.

VIII. CLASSROOM DESIGN

A. DISCUSSION

Now that several project supporting modules have been completed, the methods of presenting the software must be considered. The hardware advances of today allow for a variety of classroom presentation techniques. The classroom layout can also be modified to support calculus instruction by computer. Taken in combination, hardware improvements along with classroom layout changes can greatly enhance the instruction of calculus.

B. HARDWARE

In order to operate the project in the classroom several hardware needs arise. All designs require the classroom be equipped with a computing platform. The classroom must also support the display of video output at sufficient size for all students to see. The video display and the computer are the only devices required to support project use. However, to take advantage of all the design features of the program, several additional hardware items should be purchased. The system, either in its base form or with all the peripherals, will operate in any classroom environment.

It is best to operate the software with portable computers. Portability can be achieved by placing a desk top computer on a cart or, even more ideally, by using a laptop computer. The requirements for the desk top computer are outlined in Chapter IV. A laptop computer configuration has some additional requirements. Primarily, it must have sufficient computing power as well as the necessary video display capabilities. Keeping in mind the

ever changing computer industry, the computers recommended for use at the time of writing are listed in Table 4. This table reflects the systems recommended for both the personal computer and the Macintosh platforms.

TABLE 4 - PORTABLE COMPUTER CONFIGURATION

Computer	Processor Speed	Video Output
IBM Lap top	486-33 MHz	VGA
Powerbook Family	68030	RGB Standard

In selecting a laptop computer, it is critical to ensure its video display capability can drive an external viewing device. Not all machines have this capability, as evidenced by the Macintosh Powerbook 100 and several low end IBM compatibles. While both the desk top computer and the laptop will display all elements of the project, the laptop is more desirable because it offers the user both setup flexibility and setup speed. The user needs only to carry the computer and a display device to the classroom. Moving a cart-based desktop can be cumbersome and many instructors will be discouraged from use if required to push a computer from room to room. Therefore, laptop computers are highly recommended for driving the program.

The required display devices come in two distinct forms. The first is the oversized monitor. The minimum size monitor recommended is thirty-two diagonal inches. This monitor is of sufficient size for display in most 30-40 student classrooms. It offers superior image quality and can be positioned anywhere within the classroom. A monitor of this size, however, has some significant disadvantages. First, a thirty-two inch monitor is not easily

49

portable. (For purposes of this project, we experimented with a cart-based model and found it was very unwieldy to move.) Second, the user would have to allow a minimum of twenty minutes before class to move and set up a monitor of this size. While it is clearly an option for display purposes, it is not the most desirable one.

The second type of display device is the liquid crystal display panel, or LCD. An LCD is the most preferred method of display. The display panel is portable and can be moved easily from room to room. The setup of the LCD is simple and can be done quickly. The user needs only to place the panel on an overhead display and then hook up the computer system. A disadvantage of the liquid crystal display is that it is tied to an overhead projector and screen. It can be displayed only where these items exist in the classroom. Another disadvantage is brought out during actual use. Because the display is projector based, it is often necessary to darken the classroom to optimize the image quality. This, in turn, interferes with the students' taking notes. Finding an optimal lighting balance maybe difficult, but not impossible. The problem might also be solved by purchasing high light output overheads that show detailed displays under nearly all lighting conditions. At any rate, lighting is a potential problem that must be addressed before using the project. In sum, the project requires a display device that provides a balance between visibility and portability.

The addition of a laser pointer and laser recognizing device can greatly enhance the presentation. In the ideal configuration, the program will be used in a classroom with some

reduced lighting. The instructor will often need to highlight items of interest on the screen. The laser pointer is an excellent tool for accomplishing this task. A laser pointer operates by focusing a beam of light on the desired area of the display. The range of the standard laser far exceeds the dimensions of most classrooms, so the instructor can move freely about the classroom while using the pointer. Recent technology has taken the laser pointer idea one step further, in the form of laser recognizing devices. These devices are placed in conjunction with the liquid crystal display panel, and recognize the laser pointer's position on the screen. The device then sends the input to the computer causing it to act effectively like an extended mouse. The user can then press a second button on the laser pointer that acts as a mouse button. Thus, the user can remotely operate the program by taking advantage of the point and click design structure of the modules. The addition of this piece of hardware is especially attractive because it allows the instructor to move freely about the classroom while simultaneously operating the instruction panel. This capability of remote pointing and clicking also exists for the oversized monitor. The ability can be realized through the purchase of software that allows for the recognition. In either case, the program's structure allows hands-off manipulation of most structures. Thus, the user has many options for controlling the program.

The combination of the laptop computer, liquid crystal display panel, and laser device is the ideal configuration for using the project. The user needs only to transport a package the size of a briefcase to the classroom in order to execute the program. The setup time of

51

this configuration is less than two minutes, and complete operation of all components is achievable in less than five minutes. The "plug-and-play" nature of the devices of today allows for this rapid setup time. The system is especially appealing to the user who must move from classroom to classroom throughout the day. In addition, the laptop device could be assigned solely for the instructor's personal use. This habitual association allows the user to tailor the program to individual needs and gives the user added confidence in the intended functioning of the program. The total system allows a freedom of movement and design of instruction which was previously impossible.

C. SAMPLE CLASSROOM LAYOUT

The physical arrangement of the classrooms used for computer-based instruction is also important. Students must be able to see the main instruction screen and be able to read materials on their desks. In addition, the instructor must be able to interact with students while using the program. Each of these areas must be considered in setting up a room for computer-based instruction.

The placement of students in relation to the display screen is crucial. All students must be able to discern clearly all items on the screen. Their view of the screen also must be as unobstructed as possible, thus reducing movement by the student trying to see the screen. The ideal layout of the classroom to maximize the view of all students is shown in Figure 17. While it may be impossible to arrange all classrooms to this design, it is presented here as a recommendation for future classroom layouts.

52
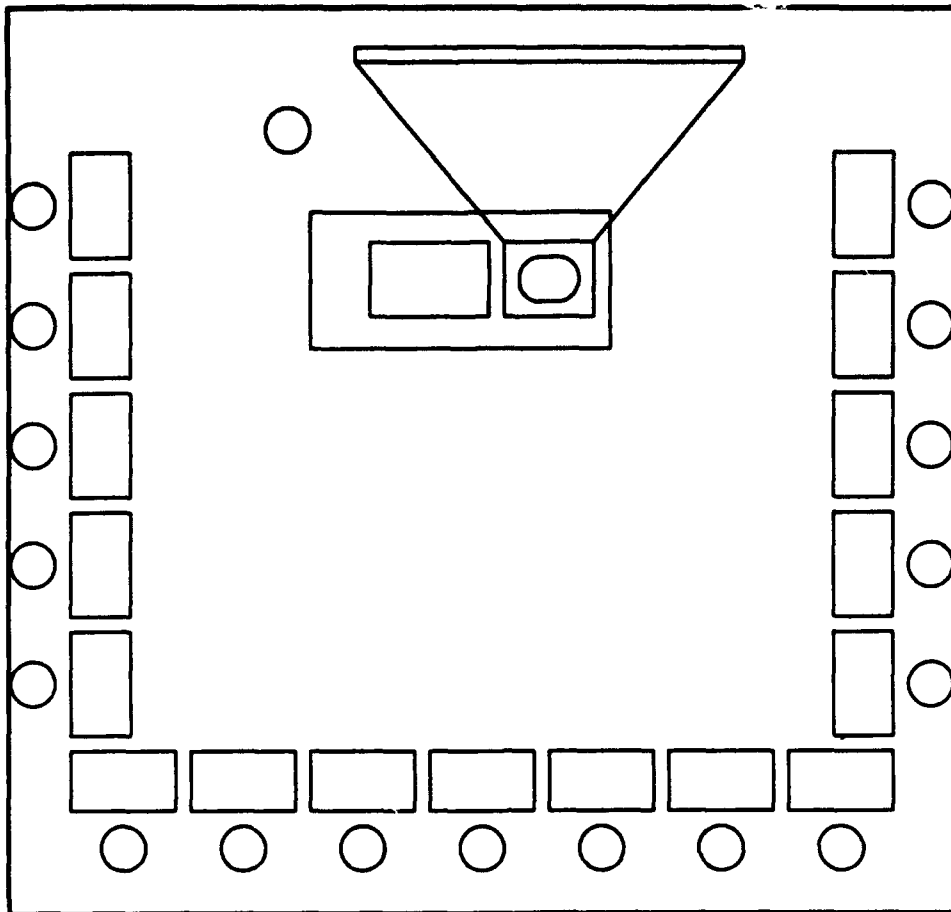
Figure 17 Classroom Layout

IX. CONCLUSION

A. GENERAL

This project explored the creation of software to assist the mathematics instructor in teaching calculus. The project took over one year to complete and the results of the work, along with the steps necessary to produce the final product, are contained in this thesis. The path that is described reflects the steps needed to produce a workable output. The goal of this thesis was to detail the steps necessary to produce the software and to identify critical areas that need be addressed. Our product offers a control structure as well as a starting point for the creation of other modules that can support instruction.

B. FURTHER RESEARCH

This project was the first step in integrating the computer into the classroom as a tool for demonstrations. During the preparation of the project, additional capabilities of the computer were discovered that would allow an even wider variety of subjects to be covered. In addition, recent technological advances, (such as the advent of CD-ROM capable computers), have opened even more avenues to explore. The advances in the computer industry, in conjunction with a more experienced programming team, will undoubtedly allow for the creation of new and innovative modules.

The modules created were initially based on a rudimentary knowledge of the operation of the computer. More advanced capabilities, such as the incorporation of movie clips or sound, were not explored. These areas offer great possibilities for future research,

and could significantly further enhance the understanding of calculus. It is important to note, however, that a good understanding of the basic operations of the computer and software is essential to using these new capabilities. Several months of training time must be invested in order to exploit these advanced features. This document might act as a road map to help decrease this training time significantly.

The advent of new technology offers directions of research. The CD-ROM allows the transport and storage of data that far exceed the capacity of the floppy disk. As the stages of the project were created, it soon became evident that computer graphics, (even when stored in their most compact form), require large amounts of memory. The files that compose this project, consisting of a controller program and twelve operational modules, far exceed the storage capacity of a single floppy disk. The CD-ROM technology allows for the storage and rapid retrieval of over fifty times the amount of material found on a conventional floppy. We note also that a device which writes to CD-ROMs is presently available, although expensive. (The price at the time of publishing this paper was approximately $4000.) The CD-ROM will soon be the standard for information distribution.

C. RECOMMENDATIONS

The use of the computer as an instructional tool is increasing. The students of tomorrow expect that instruction will take every advantage of integrating computers into the curriculum. It is essential that a university professor have at his or her disposal an easy to use tool to aid in the instruction of calculus. This project produced a product that

is multiplatform. Duality was created by producing two distinct packages for each of the current computer families. In the development phase the production of two separate systems was quite time consuming, but this should not be the case in the future. Presently, Apple's PowerPC machine will straddle the Macintosh and IBM worlds. The new machines will come standard with internal software designed to exploit programs written for both Macintosh and IBM platforms. The two families are rapidly becoming interrelated and interchangeable. The focus for designers of future modules should not be on cross platform work. Their efforts will be better spent on producing interactive modules for either system to take full advantage of the age of technology.

APPENDIX A

OPERATION OF PROGRAM MODULES

1. GENERAL COMMANDS/OBJECTS

This appendix gives a brief discussion on how to use each of the modules created for the

CalcAid program. As discussed in the main body of this thesis, the modules are laid out in

a standard fashion with structural homogeneity maintained throughout the program. As a

result, commands perform the same function all through the program. These commands,

along with the corresponding symbol when appropriate, are listed below.

<table>
<tr>
<td>File</td>
<td>This feature appears on all menu bars in all modules (see Figure 18). Clicking on this function displays a submenu with a Quit and an About... command. Clicking the Quit command starts the sequence for exiting CalcAid. Clicking About starts a routine that displays program information.</td>
</tr>
<tr>
<td>Print</td>
<td>This feature appears on all menu bars in all modules (see Figure 18). Clicking on this function displays a submenu with a Print Window command. Clicking on the Print Window command allows the user to print various slides of a module.</td>
</tr>
<tr>
<td></td>
<td>This button returns the user to the previous menu.</td>
</tr>
<tr>
<td></td>
<td>This button sends the user to the next slide.</td>
</tr>
<tr>
<td></td>
<td>This button returns the user to the previous slide.</td>
</tr>
</table>

This button begins an animation sequence.

This button plots graphs on a screen axis.

This button erases all graphs on a screen axis.

2. EQUATION CONVENTIONS

Several modules require the user to enter equations or mathematical formulas. These equations and formulas must be entered in a standard ClearTalk syntax. This syntax is easy to follow, and is similar to many other mathematical programs. A brief legend of symbols is shown below in Table 5.

TABLE 5 - CLEARTALK EQUATION SYNTAX

Symbol	Definition
+	Add
-	Subtract
*	Multiply: $3*4 = 3 \times 4$
/	Divide: $3/4 = 3 \div 4$
^	Raise to a power: $3\wedge 4 = 3^4$
cos(*angle in radians*)	Cosine of the angle: $\cos(\pi) = -1$
exp(*number*)	The natural (base-e) exponential of the number: $\exp(2) = e^2 = 7.3891$
in (*number*)	The natural (base-e) logarithm of the number: $\ln(1) = 0$
sin(*angle in radians*)	Sine of the angle: $\sin(\pi) = 0$
sqrt(*number*)	Square root of the number: $\text{sqrt}(4) = \sqrt{4} = 2$

58

3. POLAR PLOTS

This module graphs all polar equations, and comes in both a static and dynamic mode. To reach this module, click on the **Plane Curves and Polar Coordinates** entry on the main menu screen. Next, click on the **Polar Graphing** entry on the Plane Curves and Polar Coordinates submenu. This will bring up the Polar Plots module, as shown in Figure 18.
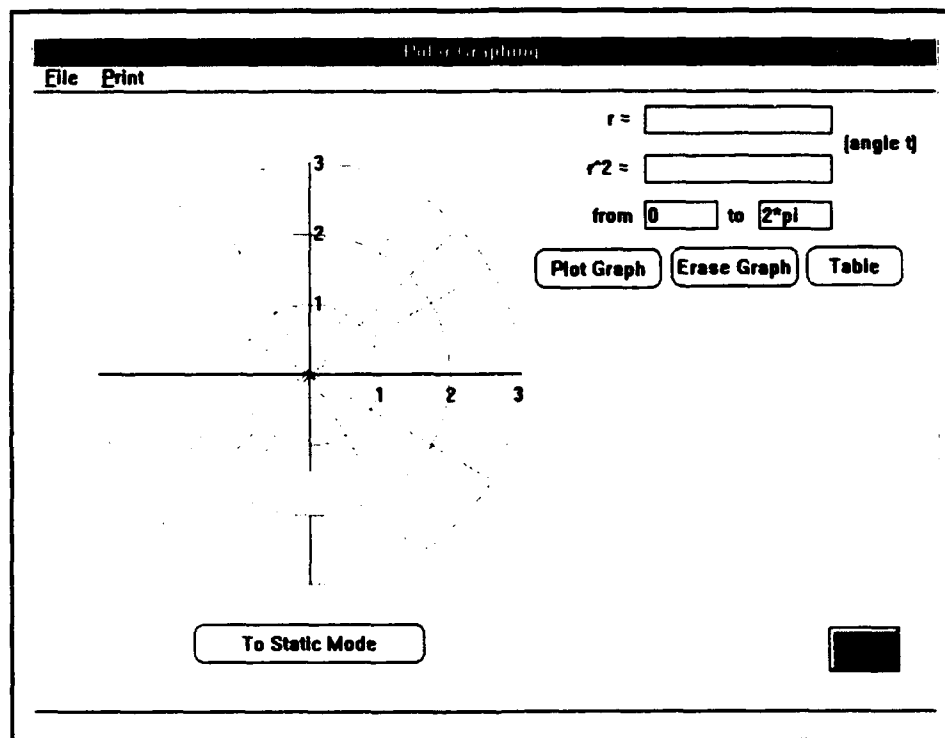
Figure 18 Dynamic Polar Plots

Equations are entered in either the r= and the r^2= fields located in the top right portion of the screen. Equations must be entered in a standard ClearTalk syntax as outlined in Table 5. The variable used in the polar equations for either the *number* or *angle in radians* entry is "t". For example, the equation for a cardioid would be entered in the r = ☐ as **1-cos(t)**.

Similarly, r= [t^2] returns a spiral. If r^2= [4*sin(2*t)] is entered, a lemniscate will be plotted on the screen.

The **from** and **to** fields on the screen are used to enter a plot start angle and a plot end angle, respectively. Again, these angles must be in radians, and entered using the ClearTalk syntax.

After entering an equation and a **from** and **to** angle, push **Plot Graph** to graph the equation. Up to four equations may be plotted in different colors. If the value of the function exceeds the axis scale, the axis will automatically readjust. The **Table** button displays a table showing the values of the equation and standard angle intervals. and then toggles to a **Hide** button. To hide the table, simply click on the **Hide** button.

The **To Static Mode** button brings up the Static Mode screen, shown in Figure 19. This screen has a list of twelve predrawn polar equation options. To display any of the given curves, click on the entry for that curve. All twelve curves, or any combination of them, may be plotted together. The **To Dynamic Mode** returns to the dynamic Polar Plots screen.
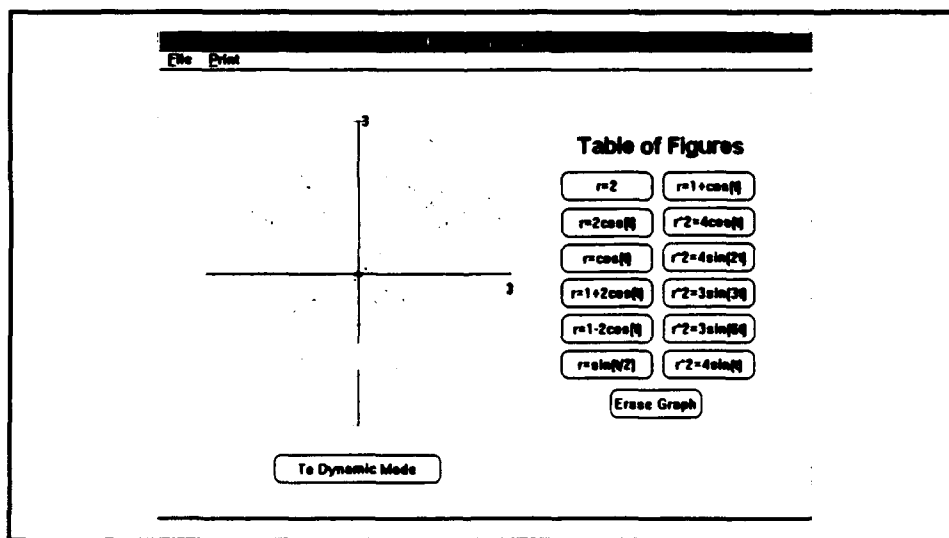
Figure 19 Static Polar Plots

60

4. SINE FUNCTION

To reach this module, begin by pressing the **Transcendental Functions** button on the main menu screen. This brings up the **Transcendental Function Submenu.** Press the **Sine Function** button, and the Sine Function module screen shown in Figure 20 will appear.
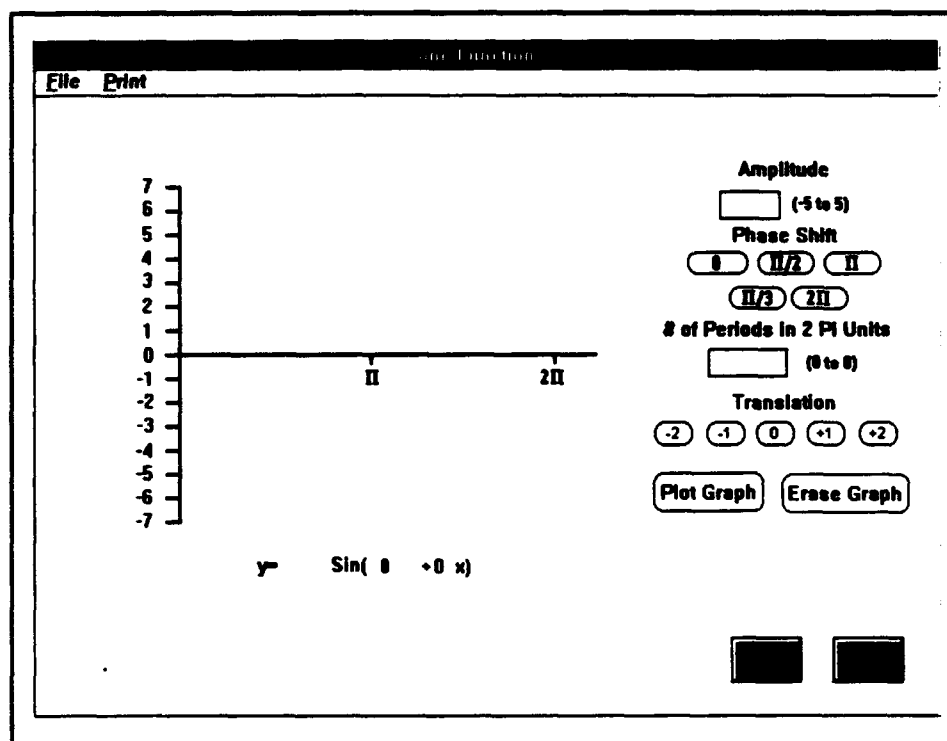
Figure 20 Dynamic Sine Function

This module was designed to facilitate the understanding of the general sine curve by enabling the user to plot the sine function with adjusted parameters. The user may vary the parameters as indicated by the general sine function of the form $f(x) = A \sin [B(x-C)] + D$, where A is the amplitude, B is the number of periods desired in a 2π interval, C is the phase shift, and D is the vertical translation. To change the amplitude, enter a value between -5

61

and 5 in the **Amplitude** ☐. To change the phase shift, click on one of the five buttons located under the word **Phase Shift**. To change the number of periods in a 2π interval, enter a value between 0 and 8 in the **# of Periods in 2 Pi Units** ☐. The five buttons under the **Translation** heading change the vertical translation of the equation.

The **Static** button brings up the Sine Function Static Mode screen, shown in Figure 21. This screen has a list of twelve predrawn sine equation options. To display any of the listed functions, click on the entry for that function. All twelve curves, or any combination of them, may be plotted together.
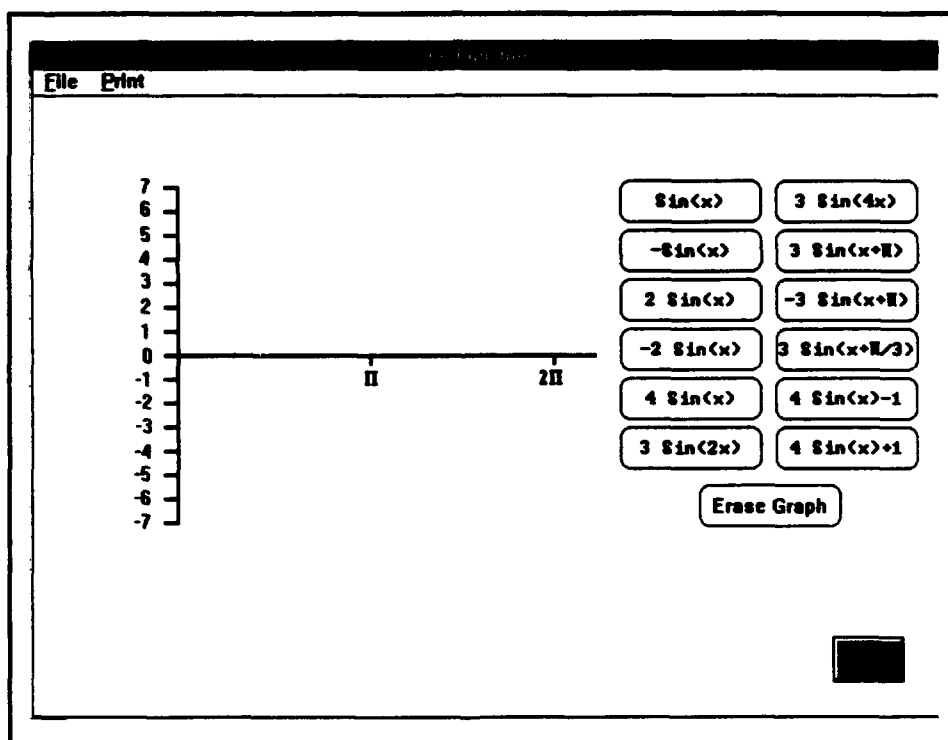
Figure 21 Static Sine Function

5. TRAPEZOIDAL RULE

To reach the Trapezoidal Rule module, choose the **Integration** button from the main menu. Next, pick the **Trapezoidal Rule** button from the Integration submenu. This calls up the first Trapezoidal Rule screen, shown in Figure 22. The first four slides of this module outline the trapezoidal rule, and contain only standard buttons as listed in section 1 of this appendix. Slide 5, as shown in Figure 23, requires some explanation. There are six subinterval sizes listed in the upper right hand portion of the screen. To partition the given curve into one of these standard sizes, click on the corresponding button. A graph of the curve with the given subinterval will appear on the screen. In addition, the area as calculated using the trapezoidal rule method will appear in the **Approximation** = ☐. The actual area, as determined by integration, is listed above this box. If an integer is typed in the $n=$ ☐, the trapezoidal area calculated with that integer subinterval will appear in the **Approximation** = ☐.
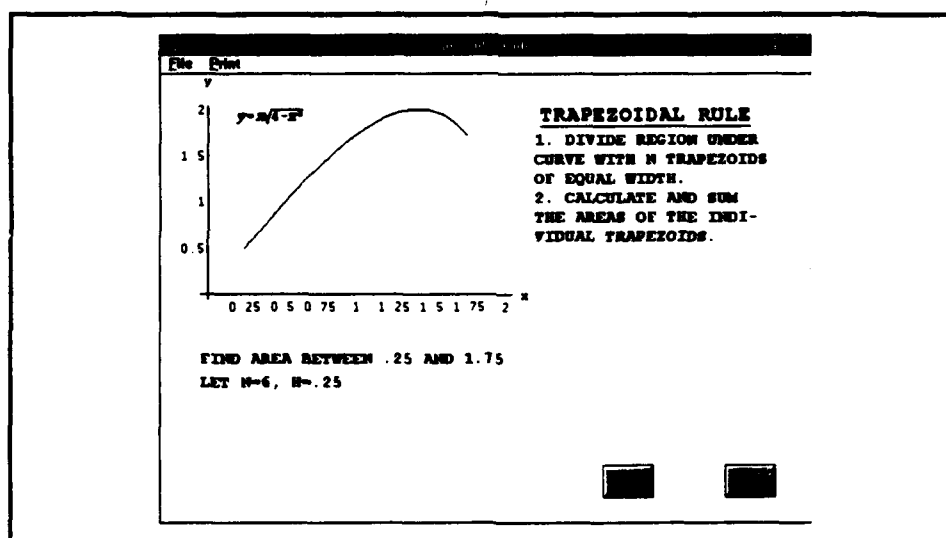
Figure 22 Trapezoidal Rule Slide 1

63
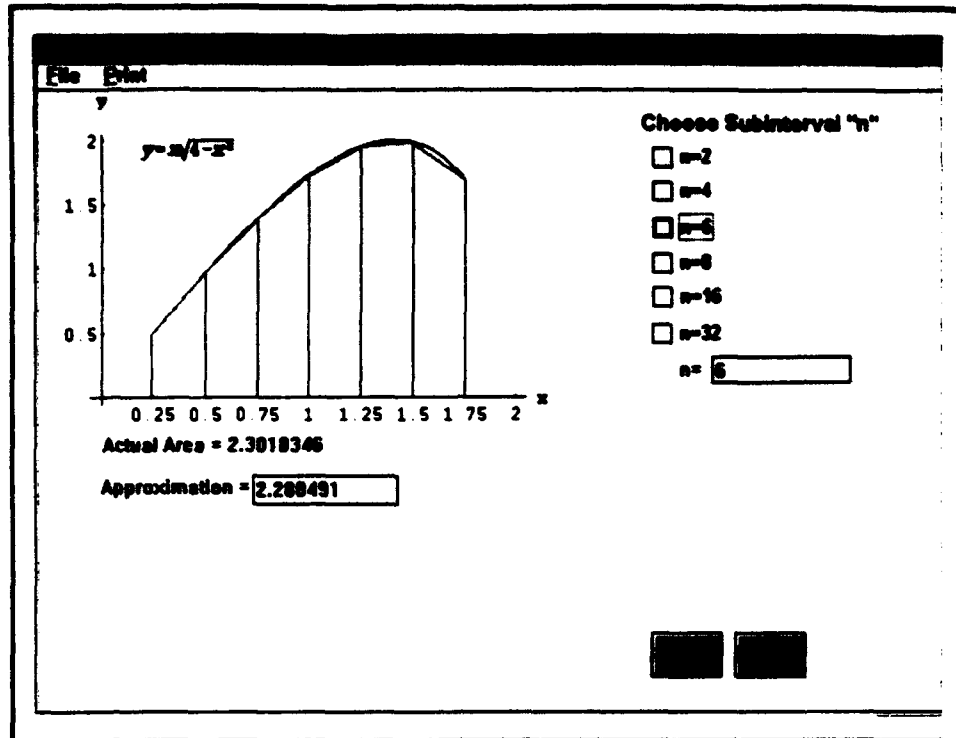
Figure 23 Trapezoidal Rule Slide 5

6. PARAMETRIC EQUATIONS

The parametric equation module consists of two submodules entitled Wheel Cycloid and

Square Cycloid. To reach these modules, click on the **Plane Curves and Polar Coordinates**

entry on the main menu screen. Next, click on the **Parametric Equations** entry on the Plane

Curves and Polar Coordinates submenu. This will call up another submenu, with **Wheel**

Cycloid and **Square Cycloid** as listed entries. By clicking the **Wheel Cycloid** entry, the

screen shown in Figure 24 appears. Likewise, by clicking the **Square Cycloid** entry, the

screen shown in Figure 25 appears. Both modules are initiated by clicking on the **Animation**

button.

64
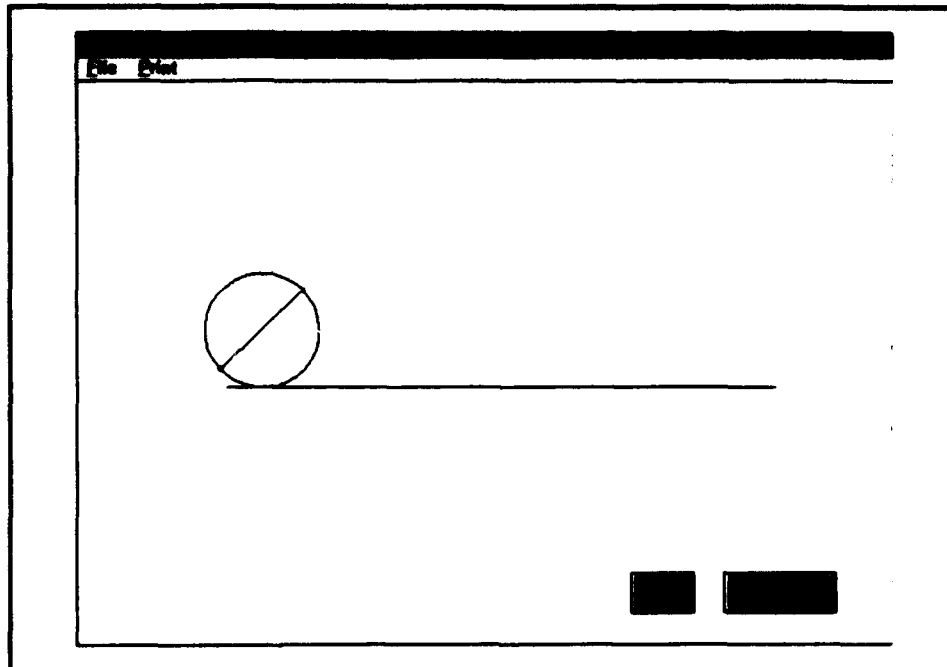
Figure 24 Wheel Cycloid
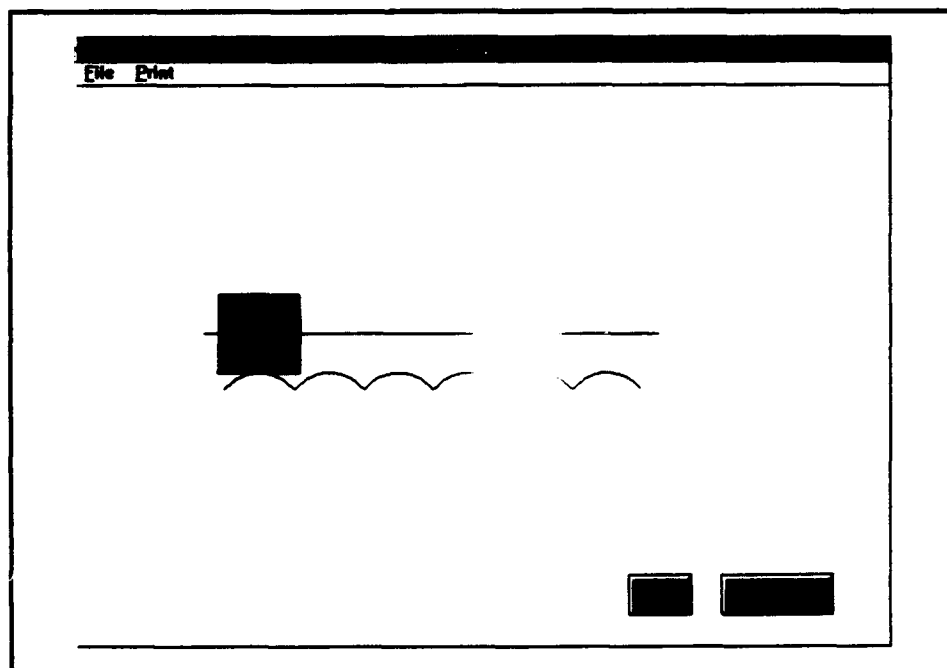
Figure 25 Square Cycloid

7. PROJECTILE MOTION

The projectile motion module illustrates the concept of ideal projectile motion. To run this module, click on the **Projectile Motion** menu on the Main Menu screen. The English Unit Projectile Motion screen will appear as in Figure 26. Calculations for this screen are based on standard English Units. The user may enter values for initial velocity, angle of elevation, initial firing position, and time. The angle of elevation may be entered in two different ways. The first way is by typing a value between 0 and 90 degrees in the **Angle of Elevation** ▢. The second way is to use the **Angle of Elevation** scroll bar. Simply click on the ends of the scroll bar to change the angle value, or physically move scroll button of the scroll bar to the desired angle value. The corresponding angle will automatically update the value in the **Angle of Elevation** ▢. The initial velocity may be changed in the same manner, using either the **Initial Velocity** ▢ or the **Initial Velocity** scroll bar. To change the firing point from the origin to another location, simply type in the coordinates of the new location in the $x0$ and $y0$ fields. Change the time of flight by typing in a new value in the t ▢. If no value is entered in this field, or if a time greater than the maximum flight time is entered, the flight time defaults to the maximum flight time.

After the flight path is plotted, output for the maximum flight time, the maximum height, the range, and the landing coordinates is presented in the corresponding ▢ locations on the lower right hand portion of the screen. The **To Metric Units** calls up the screen shown in

66

Figure 27. This screen operates in exactly the same way as the English Unit screen, except all calculations are computed using metric units.
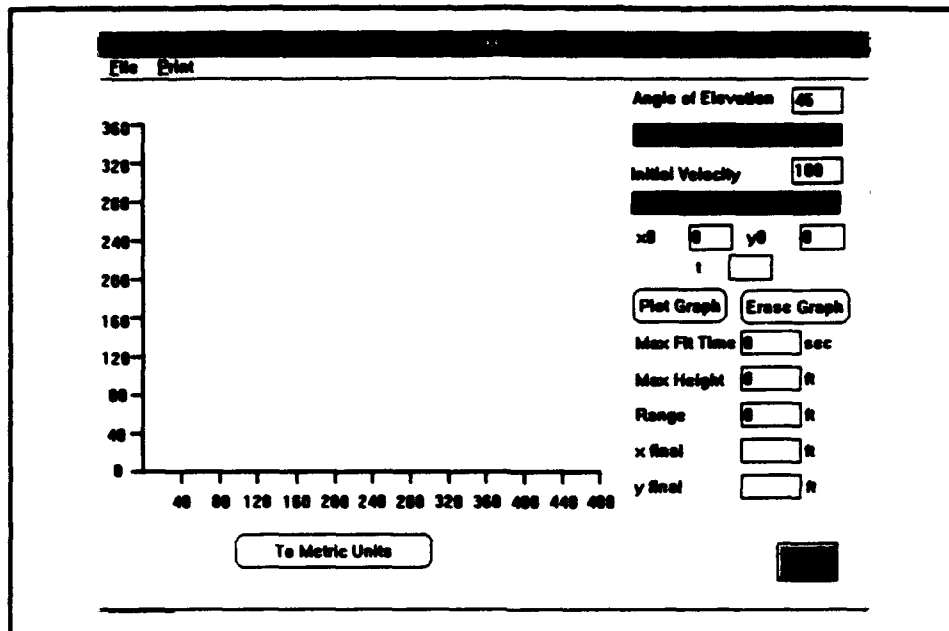
Figure 26 English Unit Projectile Motion
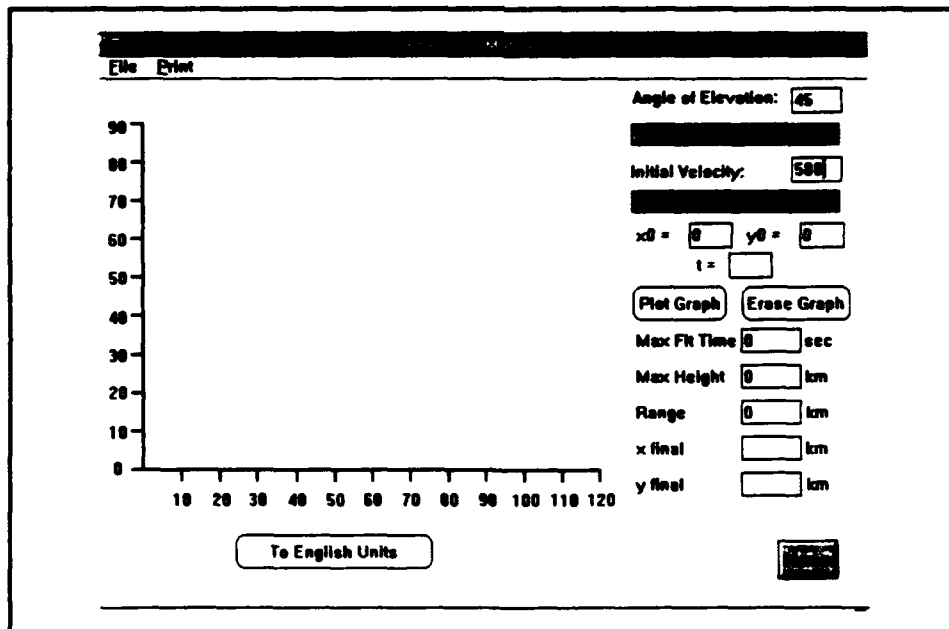
Figure 27 Metric Unit Projectile Motion

67

8. DISK METHOD

The Disk Method module gives a brief lesson on using the disk method for finding the volume of a solid of revolution. To reach the Disk Method module, choose the **Integration** button from the main menu. Next, pick the **Disk Method** button from the Integration submenu. This calls up the first Disk Method screen, shown in Figure 28. This module is completely self explanatory; the user must only click on any button to start an action. Appendix B shows a complete break down of all slides in this module.
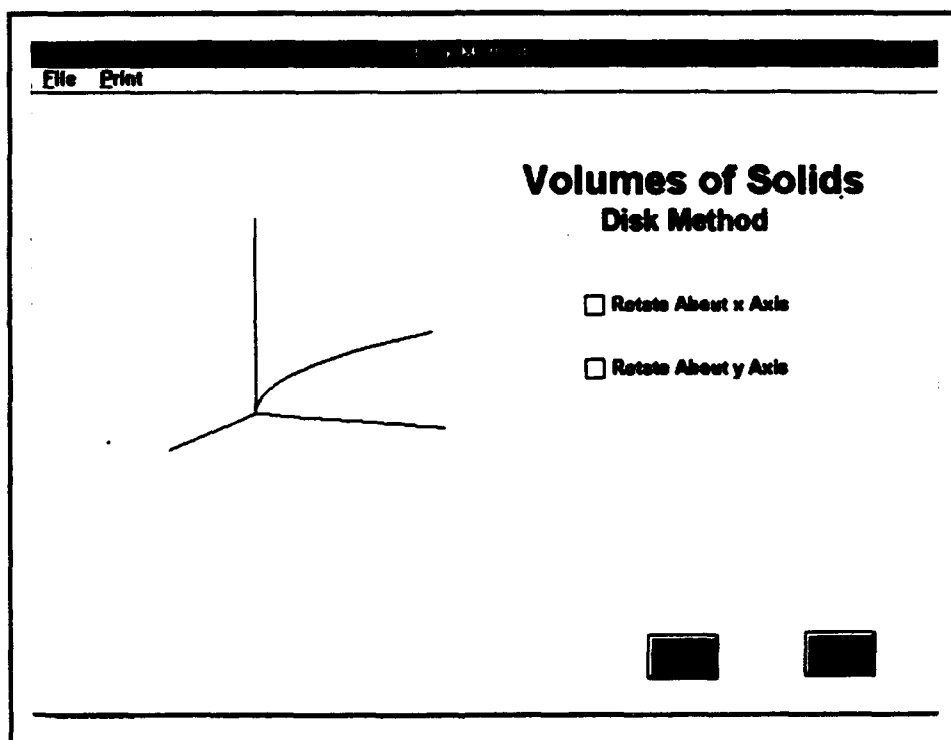
Figure 28 Disk Method

9. UNDAMPED SPRING

The Undamped Spring module interactively plots the motion of a mass on a vibrating

68

spring following Hooke's law.   The module is called up by clicking on the **Differential Equations** button on the Main menu, and then the **Undamped Spring** button on the **Differential Equations** submenu.   Figure 29 shows the Undamped Spring screen.   The Spring Constant may be changed by typing in a value in the **Spring Constant** = ☐.   The default value for this field is one.   To alter the object mass, enter a new value in the **Mass** = ☐.   Again, if no value is entered in this field, the default is one.   To change the motion time, enter a new value in the **Time** = ☐.   The module is initiated by the **Initial Displacement** scroll bar.   To give the spring an initial displacement, drag the **Initial Displacement** scroll bar button to the desired value of displacement.   The module will initiate once the scroll button is released.   After the motion of the spring is plotted, the final displacement is returned in the **Displacement** = ☐.
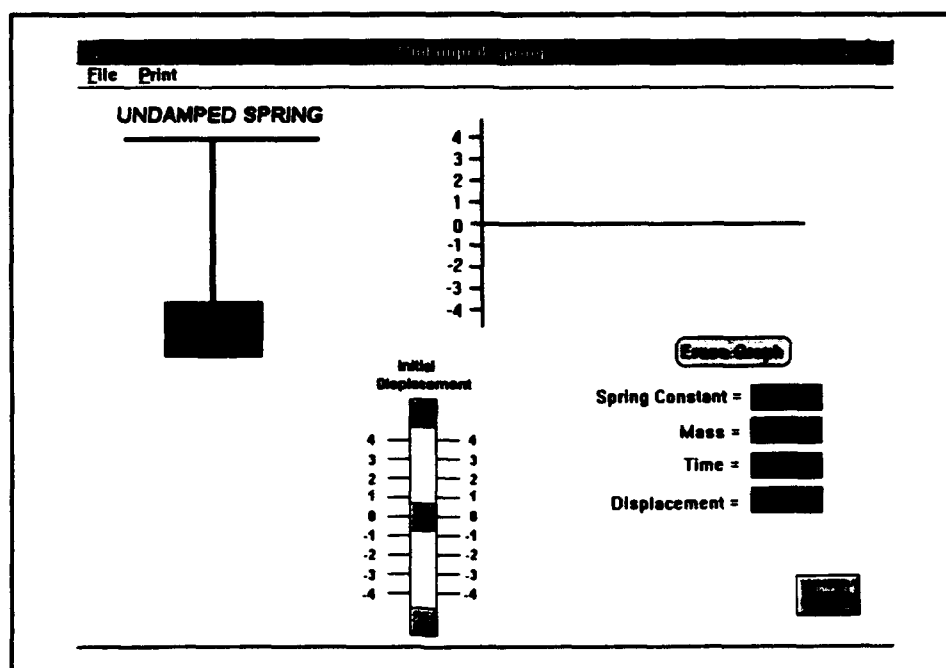
Figure 29 Undamped Spring

69

10. ABOUT...

The About... module subroutine can be reached from any slide. To activate the subroutine, first click on the File command located on all screen menu bars. Next, click on the About... command on the File submenu. The present window will disappear, and the About... window will initiate. This window is shown in Figure 30.
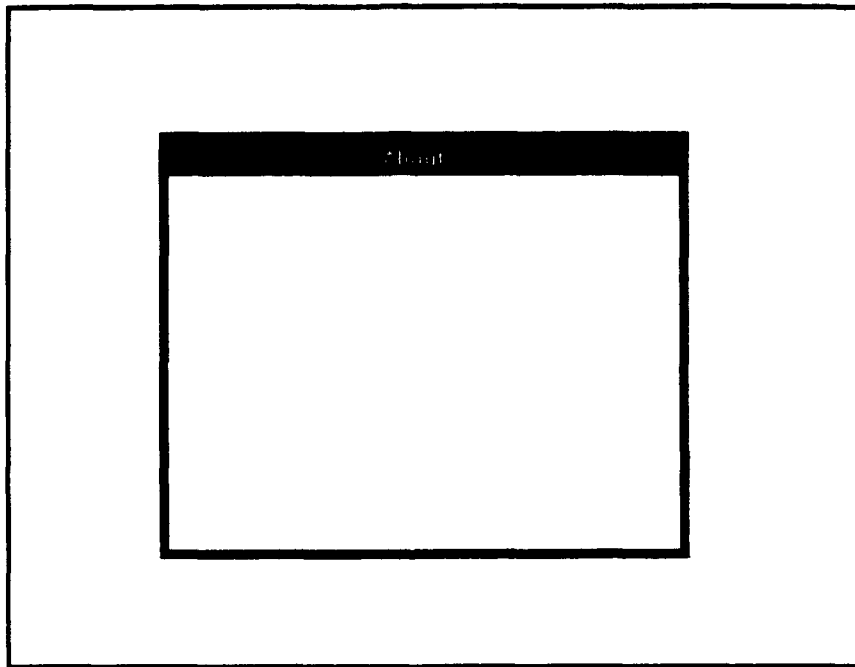
Figure 30 Initial About... Window

After this window appears, the animation sequence immediately begins. A project logo enters from the left of the window. Next, the names of the project's authors enter form the top of the window. This is followed by a field with information about the project itself. Finally, an OK button enters from the bottom of the screen. The final window is shown in Figure 31. To return to the previously hidden window, click on the **OK** button.

70
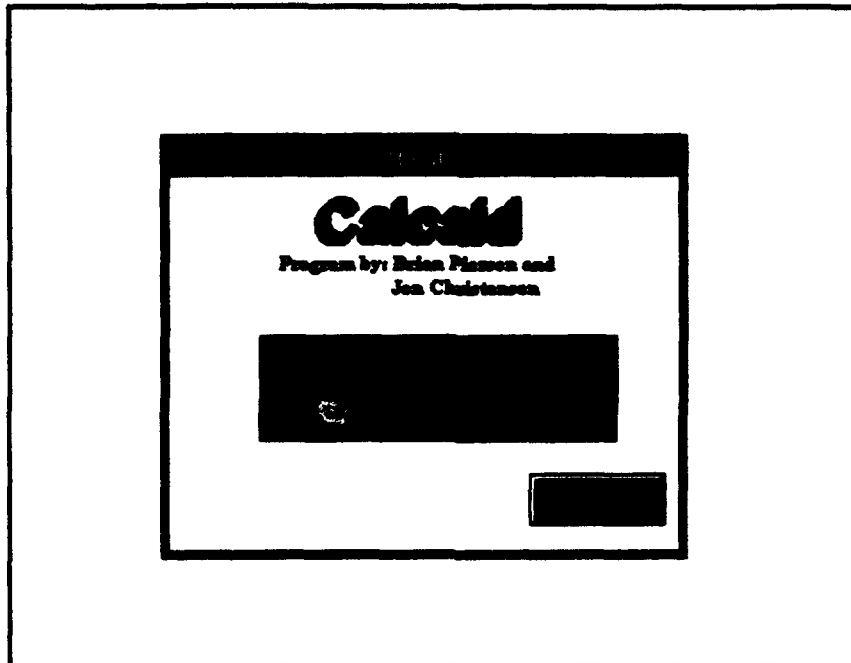
Figure 31 Final About... Screen

11. POPULATION GROWTH MODEL

The *Population Growth Module* shows the behavior of the logistical function as applied to population modeling. The module is activated by clicking on the **Dynamical Systems** on the Main menu, followed by the **Population Growth Model** button on the **Dynamical Systems** submenu. Figure 32 shows the Population Growth Model screen. To change the unrestricted growth rate, r, type in a new value in the **Growth Rate** ☐. To change the population carrying capacity, L, enter a new value in the **Carrying Capacity** ☐. Similarly, new values for the number of periods and the population initial value may be entered in their respective fields. The **Show L** button, when initiated, shows the limiting line for the population on the population graph.

71
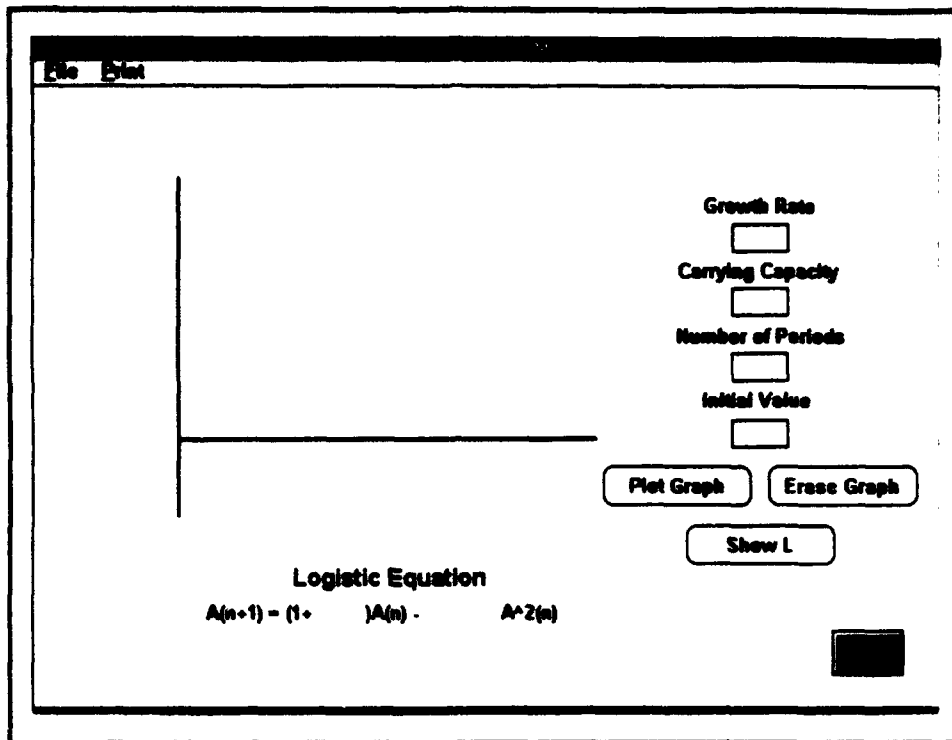
Figure 32 Population Growth Model

12. COBWEB THEORY

The Cobweb Theory module demonstrates the behavior of the logistic equation through

the graphical technique called cobwebbing.   The module is activated by clicking on the

Dynamical Systems on the Main menu, followed by the **Cobweb Theory** button on the

Dynamical Systems submenu. Figure 33 shows the Cobweb Theory Model screen. To

change the unrestricted growth rate, r, type in a new value in the **Growth Rate** ☐. To

change the population carrying capacity, L, enter a new value in the **Carrying Capacity** ☐.

Similarly, new values for the number of steps and the population initial value may be entered

72

in their respective fields. To draw the cobweb of the equation, click on the **Draw Cobweb** button. The Equilibrium Value is returned upon completion of graphing.
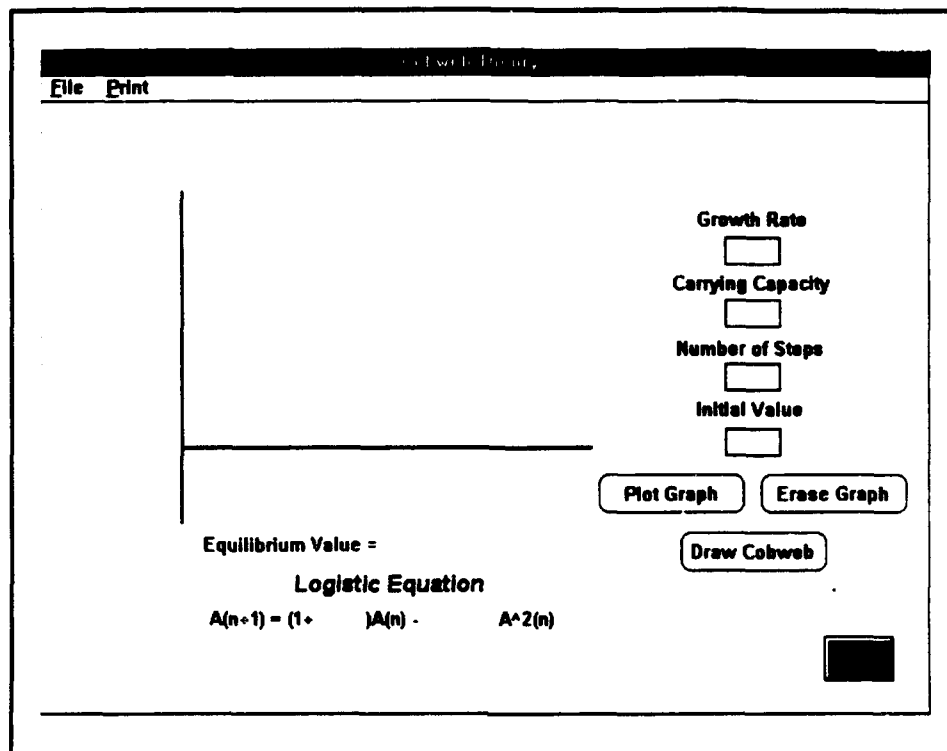
Figure 33 Cobweb Theory

13. SIMULATION MODEL

The Simulation Model module models a distribution problem typically found in a Monte Carlo type simulation. The specifics of the module are described in detail in Section K of Chapter 7. This module is triggered by clicking on the Modeling button of the Main menu. Figure 34 shows the Simulation Model screen. The only input required is the total number of students involved in the simulation. Simply enter the value in the **Simulation Number** ☐. The **Run** button activates the program. The complete breakdown of the generated

distribution is displayed graphically and in the output ☐ fields on the right side of the
screen.
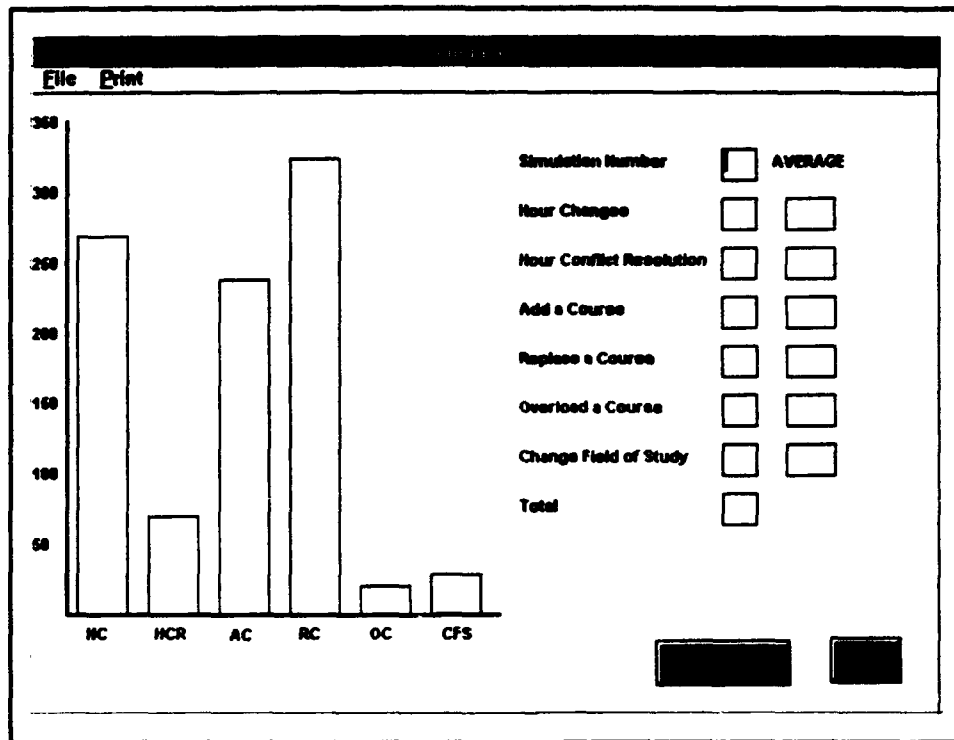
Figure 34 Simulation Model

APPENDIX B

This appendix explains displays and all of the slides that make up the Disk Method module. This module is indicative of the work that went into all other modules. The Disk Method Module is composed of 58 individual slides. The first slide, shown in figure 35, shows a curve plotted in the xyz-plane. The user is given the option to rotate the curve about the x or the y axis.
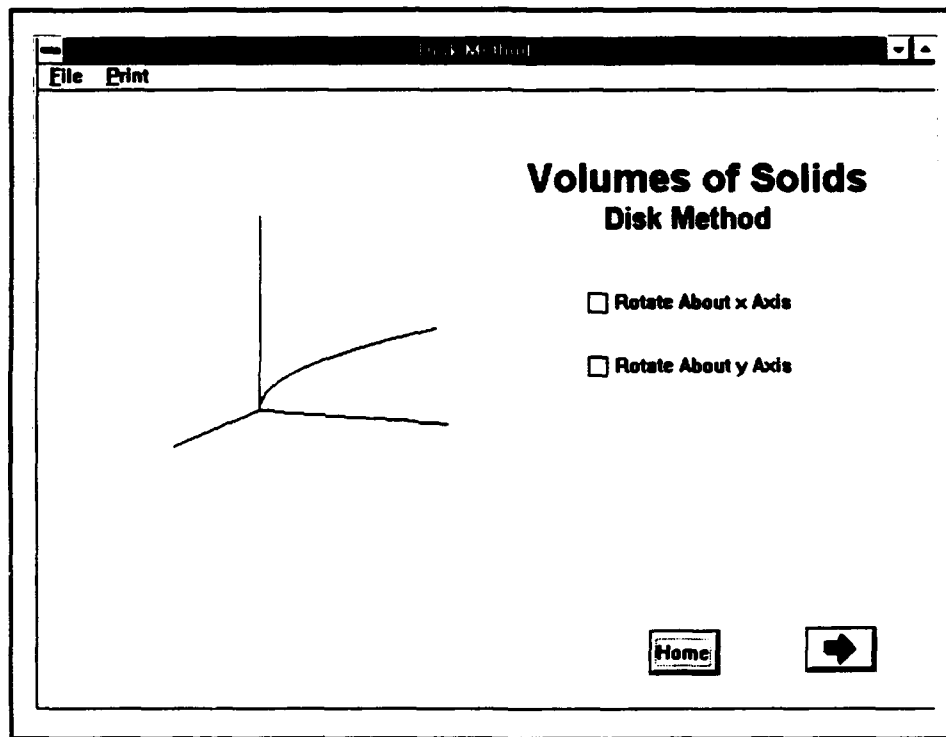
Figure 35 Initial Slide

The next 13 slides, shown in figures 36-48, display the volume generated by the curve as it is rotated about the x axis. The solids were graphed parametrically by Mathematica in $\pi/6$ increments. Because this process was exceedingly slow using the PC, the actual solids were

75

generated by the Macintosh using a Mathematica kernel running on a SUN SPARC 10

workstation.
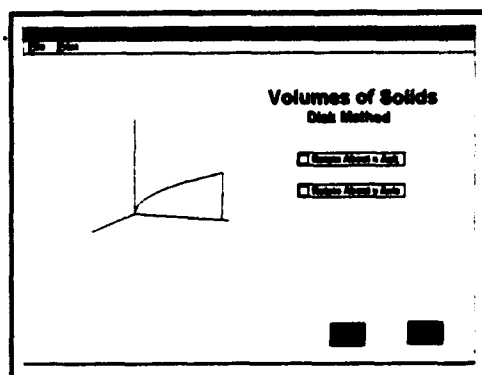
Figure 36 Slide 2
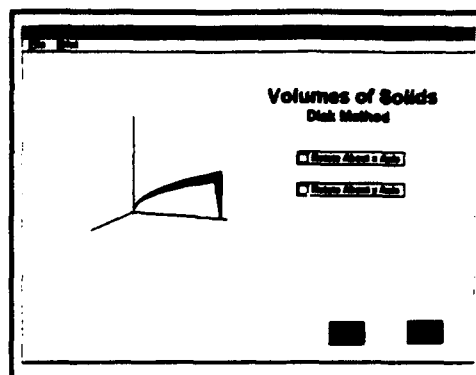
Figure 37 Slide 3
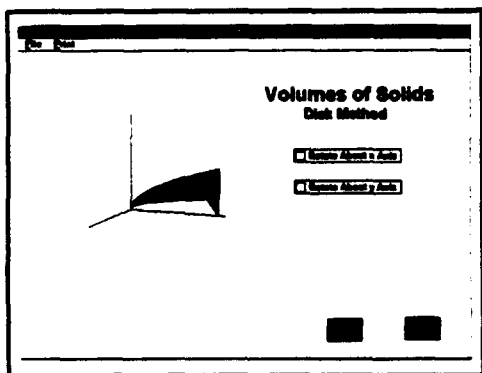
Figure 38 Slide 4
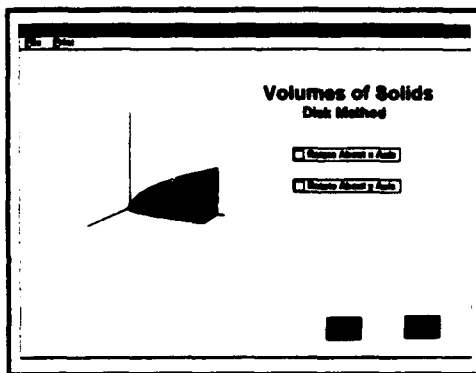
Figure 39 Slide 5
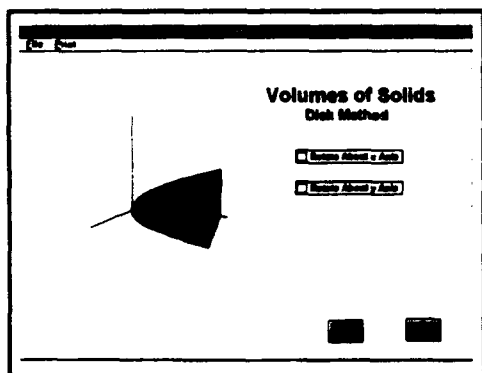
Figure 40 Slide 6
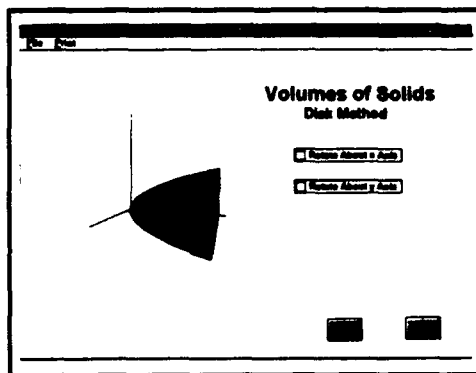
Figure 41 Slide 7
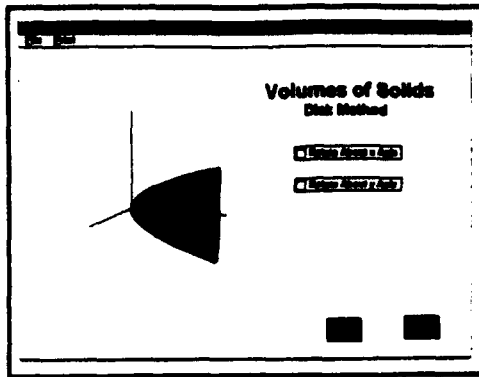
Figure 42 Slide 8
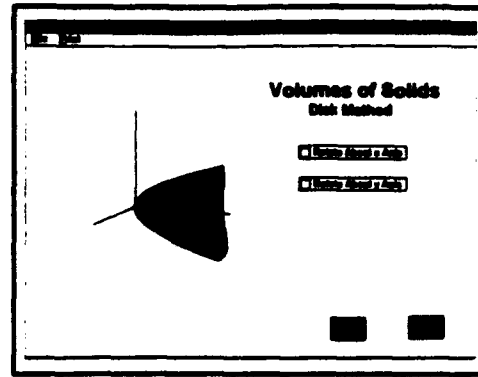
Figure 43 Slide 9
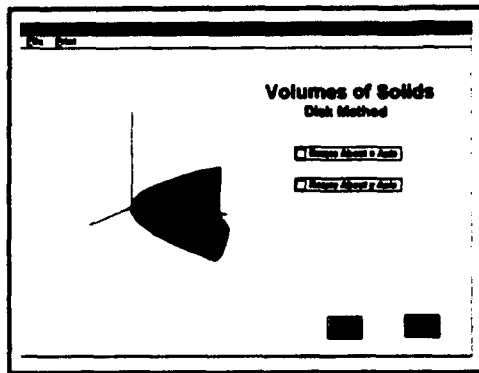
Figure 44 Slide 10
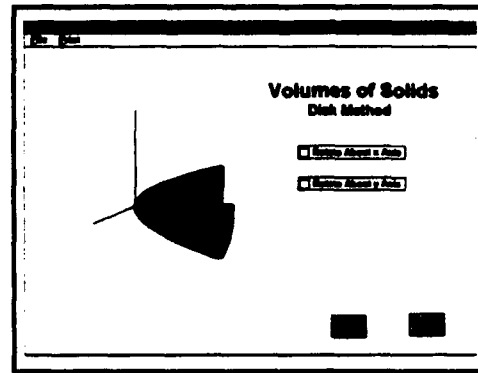
Figure 45 Slide 11
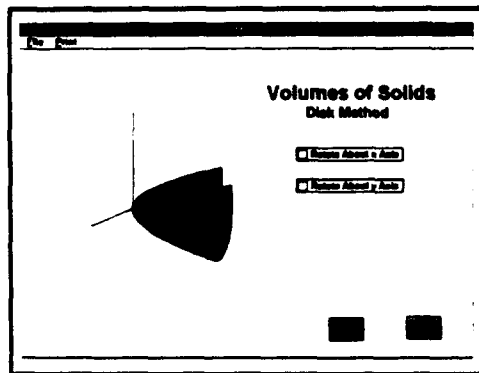
Figure 46 Slide 12

Figure 47 Slide 13

77
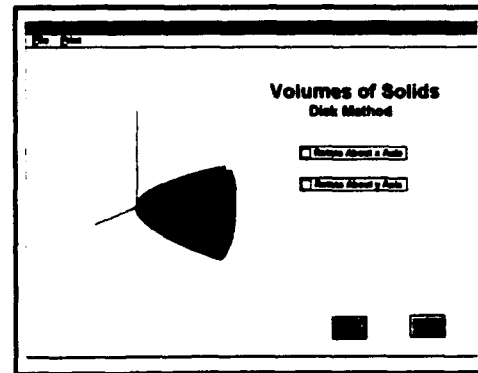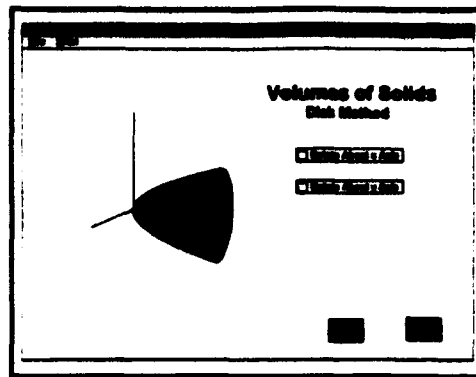
Figure 48 Slide 14

The next 13 slides, as shown in Figures 49-61, generate the solid formed by rotating the curve about the y axis.
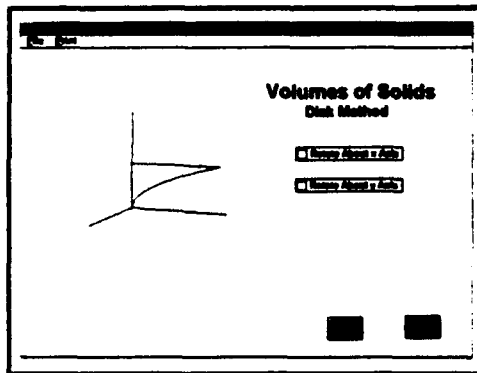
Figure 49 Slide 15
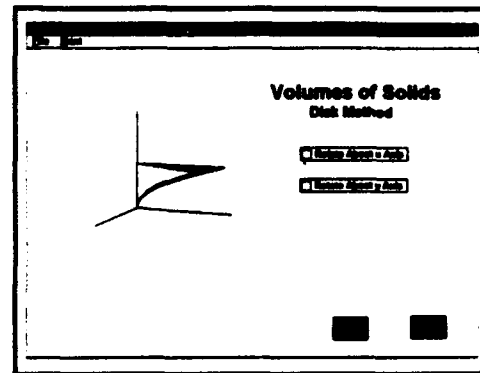
Figure 50 Slide 16
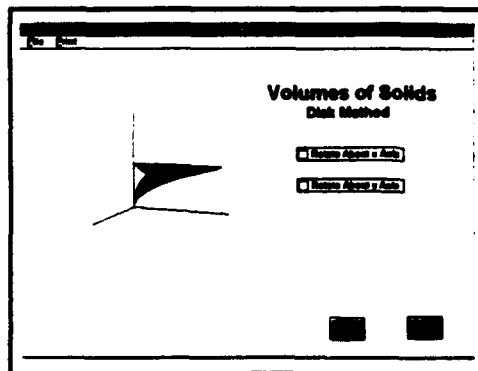
Figure 51 Slide 17
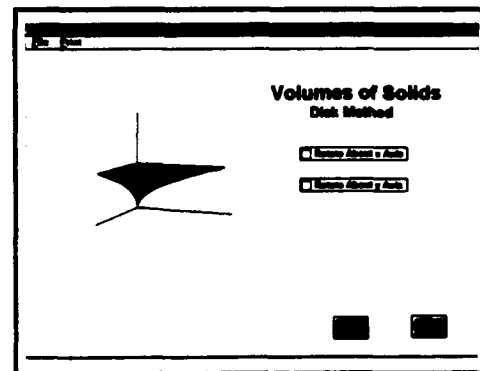
Figure 52 Slide 18

78
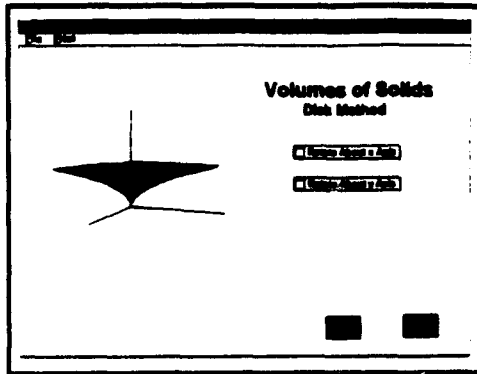
Figure 53 Slide 19
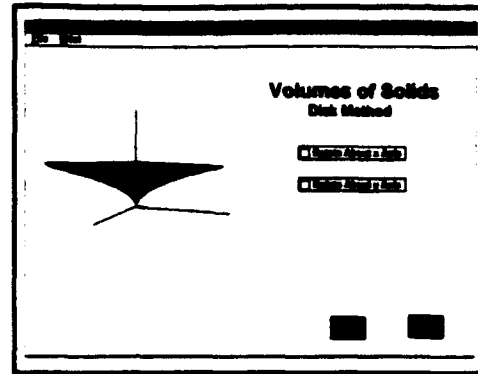
Figure 54 Slide 20
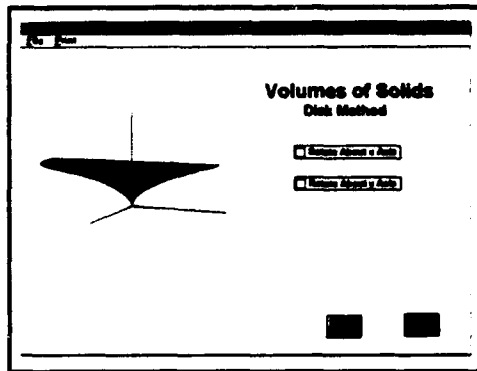
Figure 55 Slide 21
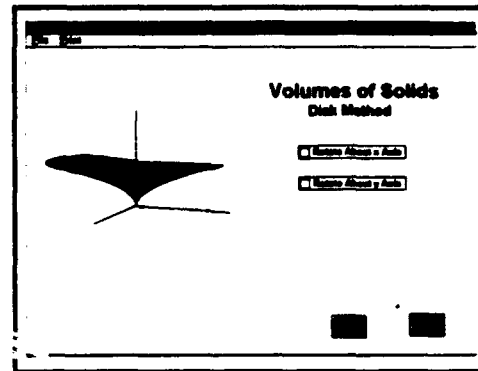
Figure 56 Slide 22
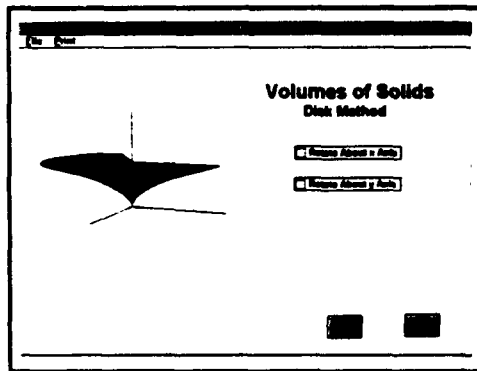
Figure 57 Slide 23
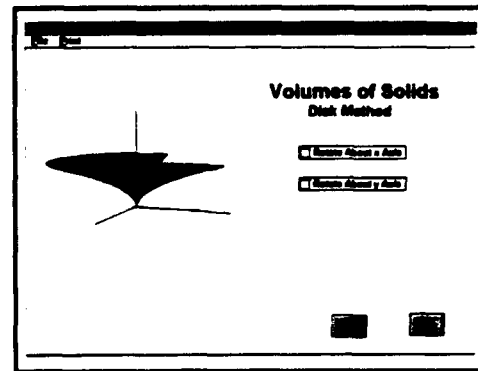
Figure 58 Slide 24
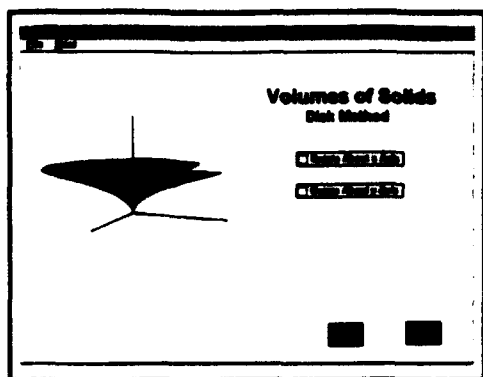
Figure 59 Slide 25
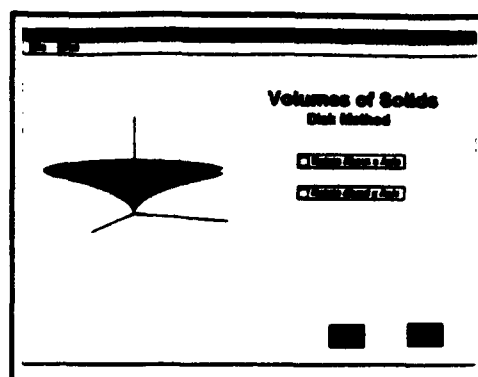
Figure 60 Slide 26
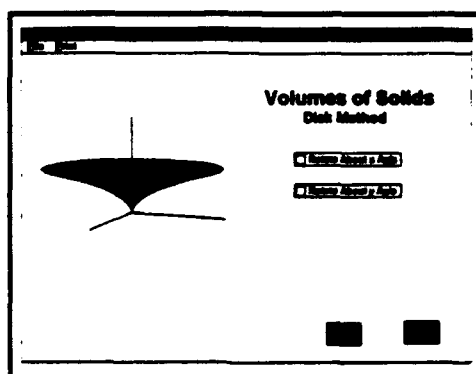
Figure 61 Slide 27

Figure 62 depicts the next slide, which explains the rectangular approximating region

concept of the disk method. The next series of slide, shown in figures 63-75, rotate the

rectangular approximating region about the x axis.
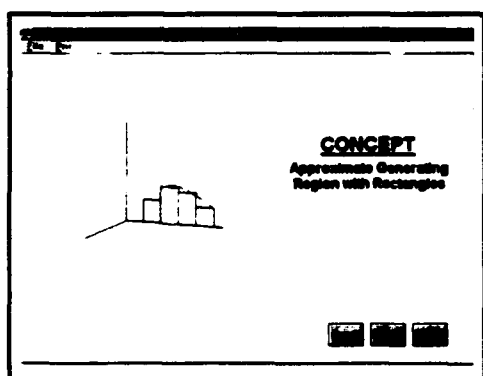
Figure 62 Slide 28

Figure 63 Slide 29
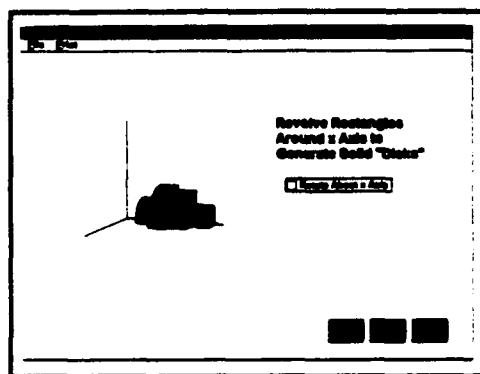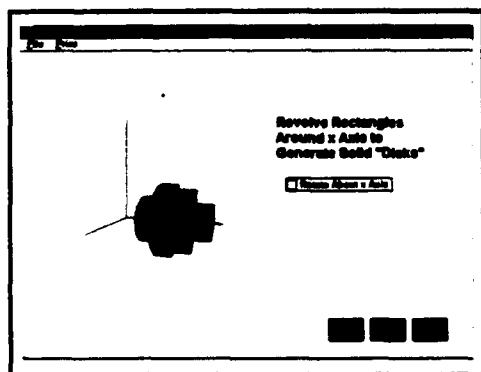
80
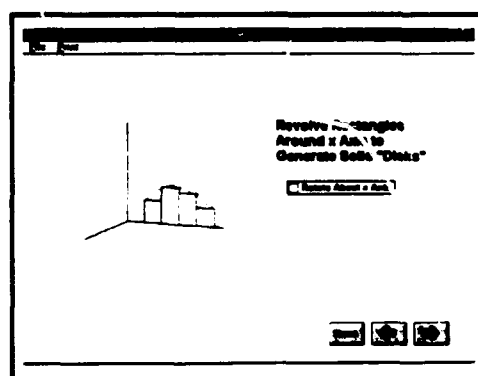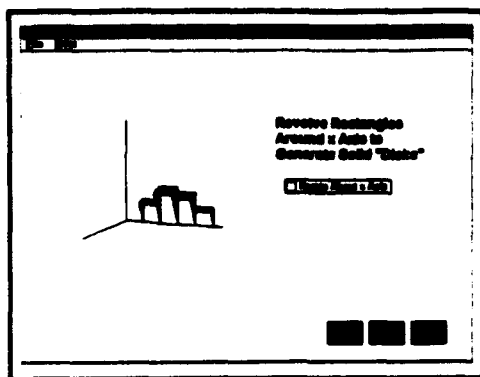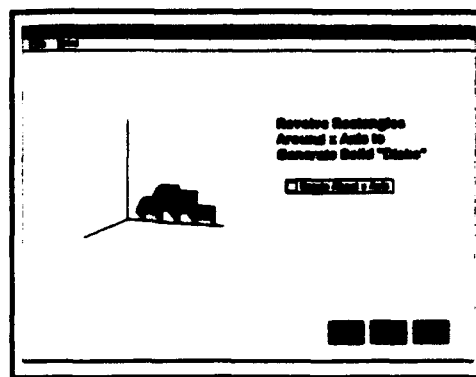
Figure 64 Slide 30

Figure 65 Slide 31

Figure 66 Slide 32
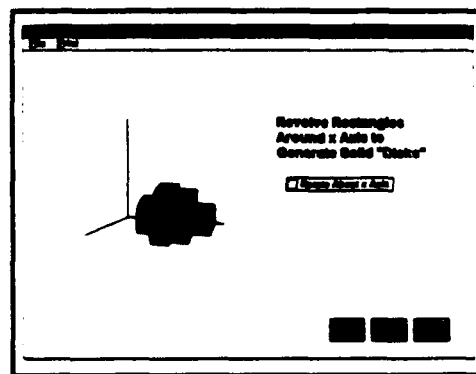
Figure 67 Slide 33

Figure 68 Slide 34

Figure 69 Slide 35

81
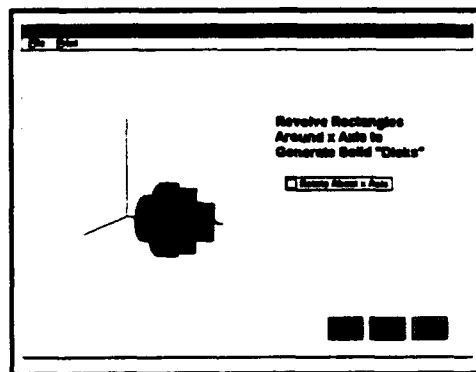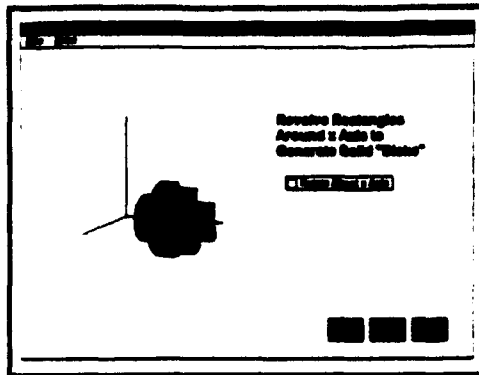
Figure 70 Slide 36
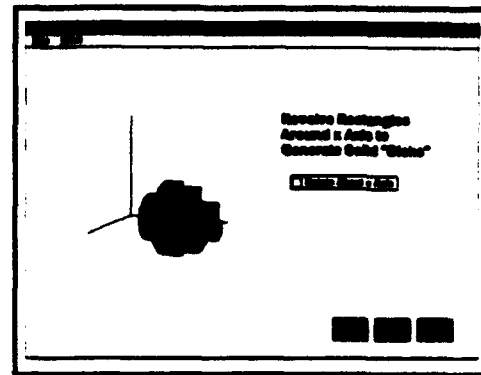
Figure 71 Slide 37
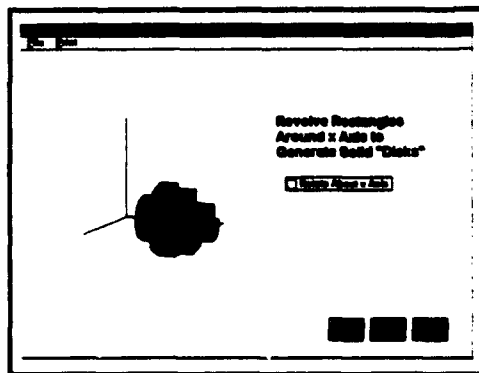
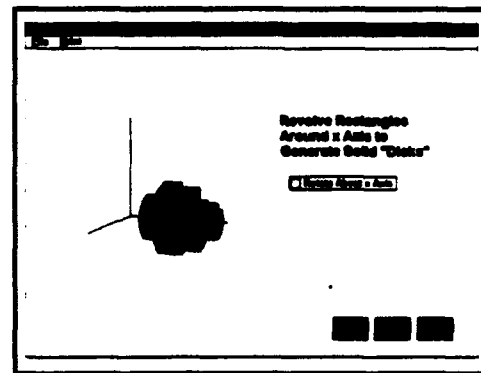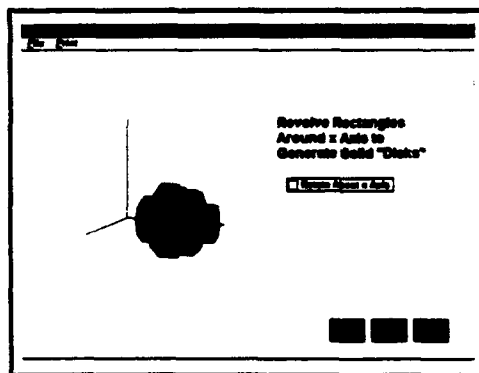**Figure 72 Slide 38**

Figure 73 Slide 39

Figure 74 Slide 40
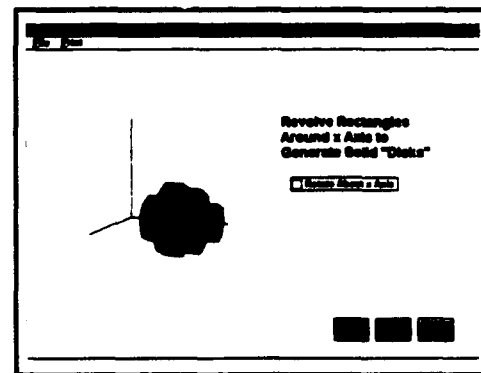
Figure 75 Slide 41

The next slide, shown in figure 76, shows the enlarged view of the "kth" disk. Figures 77 and 78 explain how to sum all disks, and show how that sum approximates the area as the disk width limit approaches zero.
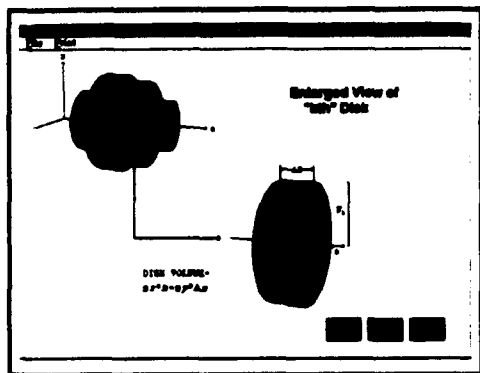
Figure 76 Slide 42
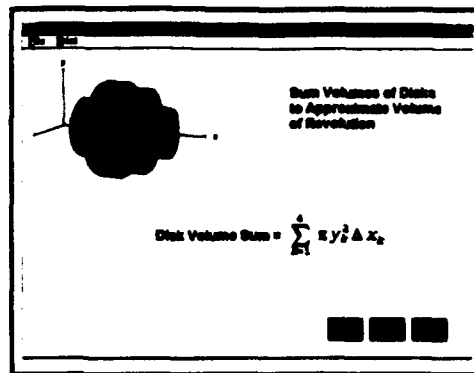
Figure 77 Slide 43
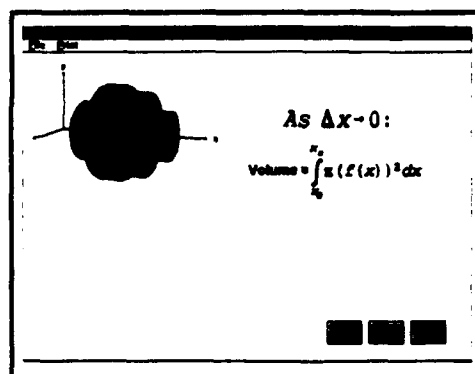
Figure 78 Slide 44

The final series of slides, displayed in Figures 79-92, show the entire process using the equation representing a quarter circle. On the final slide, the volume of the half sphere created by rotating the curve about the x axis is displayed.

83

Figure 79 Slide 45

Figure 80 Slide 46

Figure 81 Slide 47

Figure 82 Slide 48

Figure 83 Slide 49

Figure 84 Slide 50
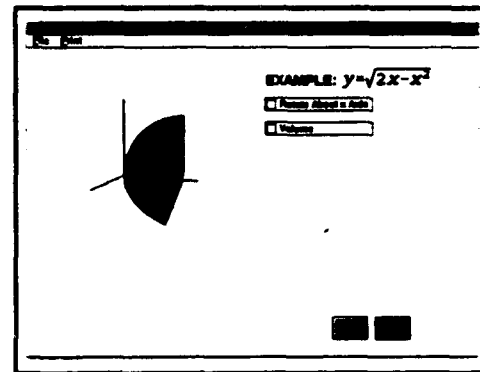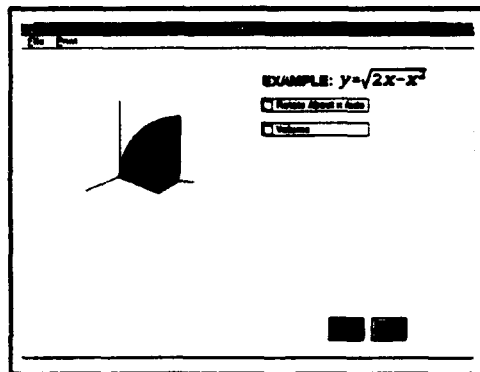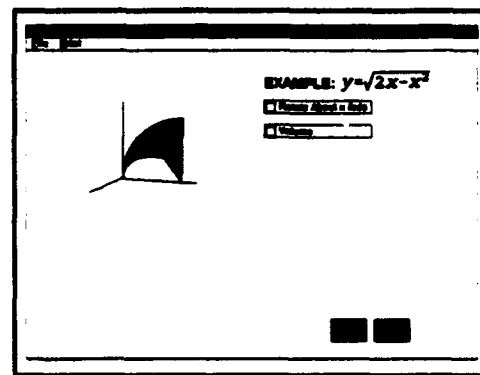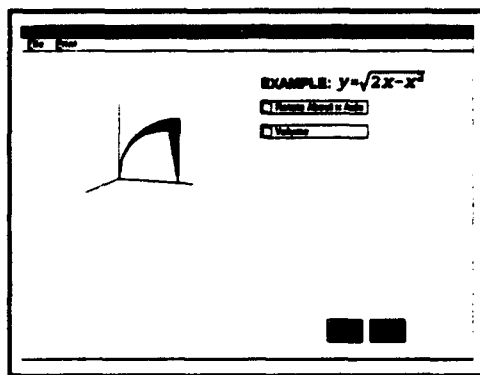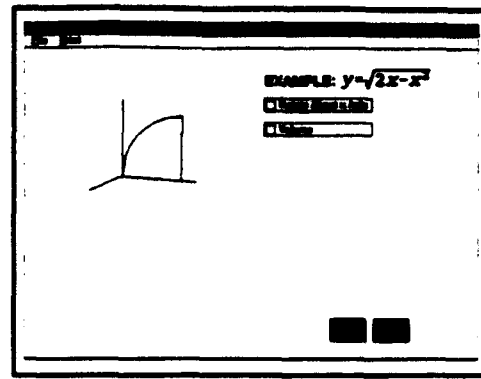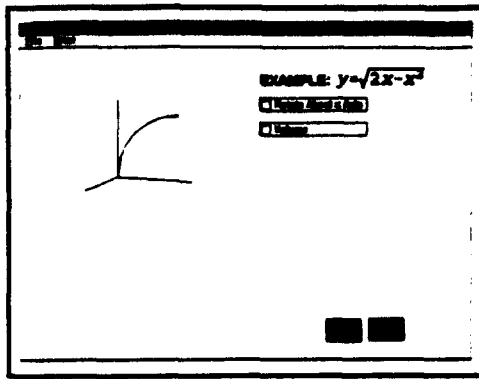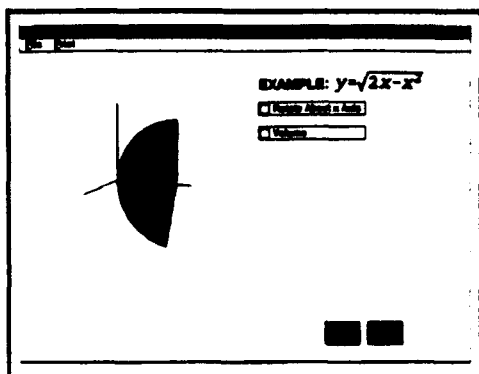
Figure 85 Slide 51
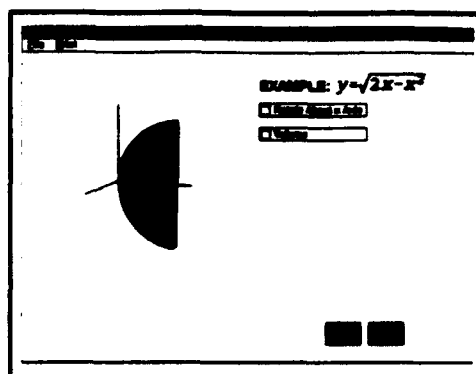
Figure 86 Slide 52
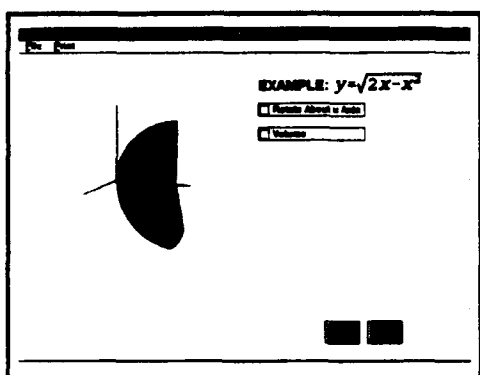
Figure 87 Slide 53
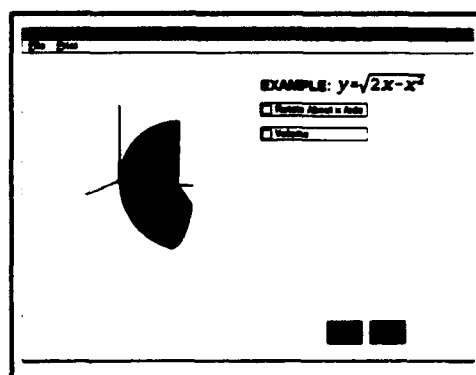
Figure 88 Slide 54
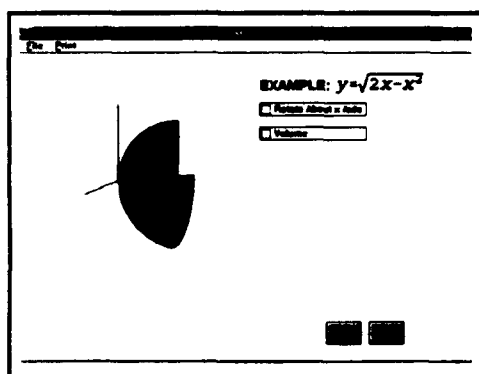
Figure 89 Slide 55
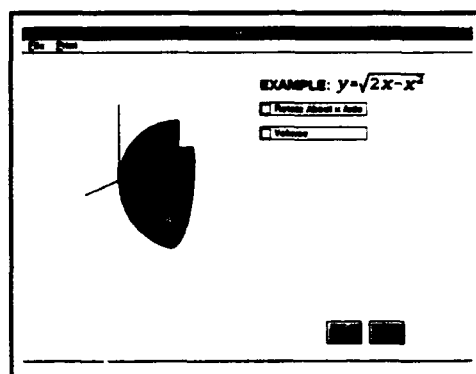
Figure 90 Slide 56
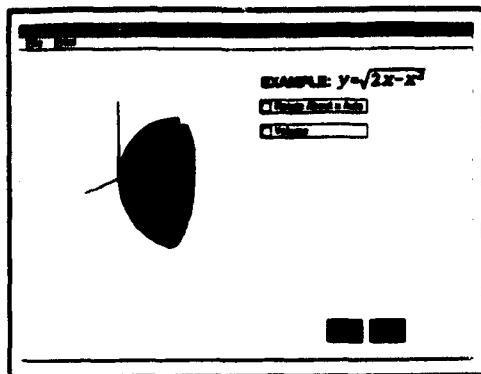
Figure 91 Slide 57
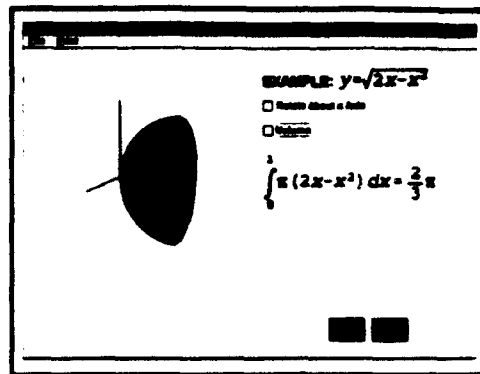
Figure 92 Slide 58

APPENDIX C

This appendix explains the ClearTalk code that drives the Undamped Spring Module. The code for this module is indicative of the code for other modules, and gives the reader a flavor for programming in the ClearTalk language. Included in the discussion are some design techniques that allow the program to function smoothly . These techniques are not inherently obvious to the first time programmer. The module screen appears below in Figure 93.
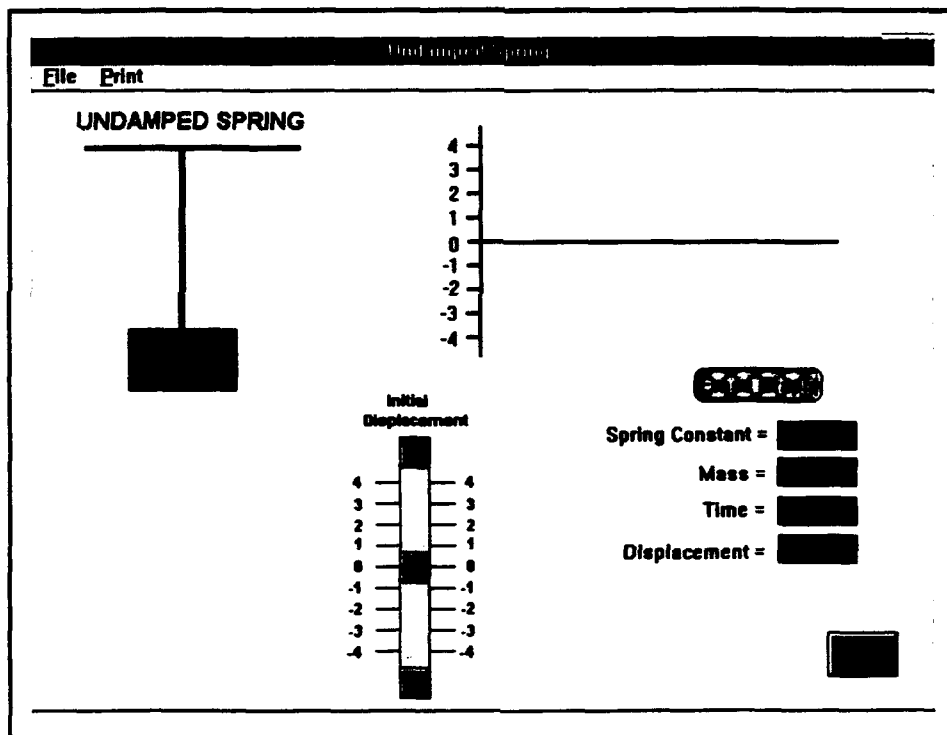
Figure 93 Undamped Spring

Each of the modules prepared for the project and the controlling structure were programmed using the script language programming technique. This unique language,

87

known as ClearTalk, offered some very desirable properties. The most significant of these properties was its ease of use. A minimum background in computer science was sufficient for understanding the language.

The starting point for the module was deciding what teaching points were to be covered. The module would demonstrate the rectilinear motion of a harmonic oscillator and plot the motion that is described in a time versus displacement graph. In order to demonstrate these concepts fully, four parameters would be allowed to vary; the spring constant (k), the mass (m), time (t) and finally the initial displacement from rest of the system (x_0). While taking these parameters in to account, a slide had to be designed that supported the display of a graph, an animated oscillator, input of four variables, and the controlling structures that allowed the module to operate. After several design proposals and layouts, the slide format shown in Figure 93 was chosen. Of particular interest is the scroll bar used for entering initial displacement. This technique was chosen because it gives a better understanding of what initial displacement means on this system.

After deciding upon an initial layout, work was initiated on the supporting code. The programming of modules is conducted in three distinct phases; the collection of input for calculations, the graphical display of the input, and finally the creation of an animation that supports understanding the demonstrated concept. By following this order, the programmer has a logical sequence in which to debug the program. This order also cuts down on errors, as mistakes are less likely to be compounded. The script that drives the program will now

be described in detail. Hopefully, this will assist future work in creating similar modules.

The majority of the script that operates the module is associated with the slide itself, and is not tied to a specific object. The code that is present in the slide level of programming consists of four subroutines. Each of the subroutine functions is essential to the proper performance of the module. The first code that a module executes is the open slide subroutine. This routine executes each time the slide is brought into operation.

```
on openSlide
    global msl,mxorg,myorg,massrec,xorg,yorg,colorcnt,coord
    set loc of foreground draw object"mass" to loc background paint field "mass
    slider"
    put the loc of background paint field "mass slider" into morg
    put item 1 of morg into mxorg
    put item 2 of morg into myorg
    put the rect of paint field "mass slider" into msl
    put the rect of foreground draw object "mass" massrec
    put the rect of paint field "graph plot" into coord
    put item 1 of coord into xorg
    put round ((item 4 of coord + item 2 of coord)/2) into yorg
    set the brush to 8
    put 0 into colorcnt
end openslide
```

The subroutine initializes the global variables used by the program and initializes the location of screen graphics. The routine then dynamically retrieves the location of various objects and places their screen locations into variables used by the program. The final command at the end of the routine sets the size of the paint brush that will be utilized throughout the module. The subroutine then terminates and returns control to the user.

89

The counterpart to the open slide routine is the close slide routine. This subroutine is executed upon module termination. The close slide routine for this module is given below.

```
on closeSlide
  lock screen
  put empty into cd fld "k"
  put 15 into cd fld "m"
  get the rect of paint field "graph plot"
  choose select tool
  drag from item 1 of it, item 2 of it to item 3 of it, item 4 of it
  doMenu clear picture
  choose browse tool
end closeSlide
```

The close slide subroutine begins by not allowing any further changes to the screen by issuing the lockscreen command. The program then empties two variable registers. The final series of commands uses the tools available in the scripting language to erase any user defined graphics produced during the use of the module.

The final subroutine on the slide level of programs is the workhorse of the module. It contains the procedures that perform the majority of the calculations associated with the module. This subroutine also controls the graphic tools used to produce plots by the module. This module's "executable" subroutine is described line by line to facilitate understanding of the code.

```
on plotGraph disp,k,m,tf
(This module activates on the command of plotGraph and utilizes the variables stored in
registers disp,k,m,and tf)
  put tf into maxX
(The amount of time the spring will oscillate is placed into maxX)
  put 0 into minX
(The minX variable is initialized to 0)
```

90

```
put 4 into maxY
```
(The maxY variable is initialized to 4)
```
put -4 into minY
```
(The minY variable is initialized to -4)
```
global msl,mxorg,myorg,massrec,xorg,yorg,colorcnt,coord
```
(The global variables listed are made available for use)

```
if colorcnt = 1 then
  set fillColor to 10
else if colorcnt = 2 then
  set fillColor to 13
else if colorcnt = 3 then
  set fillColor to 17
else if colorcnt = 4 then
  set fillColor to 15
else
  set fillColor to 1
end if
```
(This loop sets the color the program will use in plotting graphs. The loop is incremented, and plots that will overlay one another are displayed in different colors)
```
put colorcnt + 1 into colorcnt
```
(The variable colorcnt is incremented)
```
choose "brush paint tool"
```
(The graphic drawing tool is selected for use)
```
put 0 into t
```
(The time variable is initialized)
```
put .1 into incr
```
(The graphic step increment is initialized)
```
put round((item 3 of coord -item 1 of coord)/maxX) into temp1
put round((item 4 of coord -item 2 of coord)/(maxY-minY)) into temp2
```
(These commands translate world coordinates to screen coordinates)
```
put sqrt(k/m) into temp3
put round ((item 4 of msl - item 2 of msl)/(maxY - minY)) into temp4
```
(The screen location of the spring animator is initialized)

```
repeat with counter = 1 to round (maxX/incr)
```
(This loop creates the graph and moves the spring mass incrementally)
```
  put round(temp1*t) into scalxt
```
  (The x-direction screen correlation coefficient is calculated for the graph)
```
  put round(temp2*(disp*cos(temp3*t))) into scalyt
```

91

(The y-direction screen correlation coefficient is calculated for the graph)
put (xorg + scalxt) into xclick
(The screen x location is calculated for the graph plot)
put (yorg - scalyt) into yclick
(The screen x location is calculated for the graph plot)
if yclick > 37 and yclick < 176 then
    if xclick < 573 and xclick >= 319 then
        click at xclick,yclick
    end if
end if
(The if-then structure is an error check to ensure that the point to be plotted is within the display parameters)
put round(temp4*(disp*cos(temp3*t))) into scalmt
(The x-direction screen correlation coefficient is calculated for the spring mass)
set the height of btn "cover" to 360
(A hider graphic position is initialized)
set loc of foreground draw object "mass" to mxorg, myorg - scalmt
(The mass is positioned based on the calculated coordinates)
add incr to t
(The time variable is incremented for the next iteration)
set the cursor to wait
(The mouse point is set to a wait state)
end repeat

put (disp*cos(temp3*t)) into cd fld "Displacement"
(The final displacement location of the spring is calculated and placed into the card field for display to the user)
choose browse tool
(Control is returned to the user)
end plotGraph

In summation, the plotGraph subroutine dynamically grabs the screen location of a number of graphics and stores their locations by pixel into selected variables. These variables are then used by scaling equations in order to present a scaled version of the desired equation on the screen. In conjunction with this procedure, a screen graphic depicting the rectilinear motion is positioned incrementally to give the illusion of motion. Finally, the

92

discrete loop is exited and the displacement of the mass is displayed.

The program also has scripts associated with several of the on screen graphics that detail and control their function. For this module the erase graph button, displacement scroll bar, and home button all have scripts. The unique characteristics of each of these graphics warrants description to clarify how they operate within and control the module.

The erase button performs three distinct functions. It executes a routine that erases any graphs that have been created, reinitializes all variables, and resets the animation graphics.

```
on mouseUp
(Execute this routine when the button is clicked on by the mouse)
    set the cursor to wait
    (Allow no further user input)
    put "" into cd fld "k"
    (Empty the spring constant variable)
    put "" into cd fld "m"
    (Empty the mass variable)
    put 15 into cd fld "tf"
    (Put 15 into the time factor variable)
    put 0 into cd fld "Displacement"
    (Display zero as the current mass displacement)
    set the thumbPosition of scroll bar 1 to 400
    (Places the slider to the midpoint of the scroll bar)
    set the loc of foreground draw object "mass to loc of background        paint
            field "mass slider"
    (Moves the mass to the initial position)
    set the height of btn "cover" to 169
    (Sets the hider graphic that supports the animation to the initial position)
    set the bottom of btn "cover" to 360
    (Sets the hider graphic that supports the animation to the initial position)
    global colorcnt
    (Makes the global variable colorcnt accessible)
    put 0 into colorcnt
    (Resets the color counter variable)
    get the rect of paint fld "graph plot"
```

93

(Places the coordinates of the current graph into memory)
choose select tool
(Selects a graphic controlling tool)
drag from item 1 of it, item 2 of it to item 3 of it, item 4 of it
(Places a selecting box around the current image)
doMenu clear picture
(Erases the current image from the screen)
choose browse tool
(Returns control to the user)
end mouseUp

The displacement scroll bar is used to transmit the initial displacement of the spring mass

system to the program. The mouse is placed on the scroll bar and then is dragged to the

position that the user specifies. The callback script associated with the scroll bar feeds the

displacement value to the program and displaces the animated graphic to the desired location.

Upon release of the scroll bar the program is initiated and the animated motion and plots are

produced.

on mouseUp
(Routine activated by the mouse)
  global morg,mxorg,myorg,msl,massrec,coord,xorg
  (Allows the global variables to be accessed)
  put the loc of background paint field "mass slider" into morg
  (Places the graphical object's screen coordinates into a variable)
  put item 1 of morg into mxorg
  (Separates the morg variable's x coordinate)
  put item 2 of morg into myorg
  (Separates the morg variable's y coordinate)
  put the rect of background paint field "mass slider" into msl
  (Places the graphical object's screen coordinates into a variable)
  put the rect of foreground draw object "mass" into massrec
  (Places the graphical object's screen coordinates into a variable)
  put the rect of paint field "graph plot" into coord
  (Places the graphical object's screen coordinates into a variable)

94

```
put item 1 of coord into xorg
(Places the x coordinate of the graph into a variable)
put round((item 4 of coord + item 2 of coord)/2) into yorg
(Calculates the pixel size of the current graph location)
set the brush to 8
(Selects the size of the graphical drawing tool used to plot the graph)
put word 1 of cd fld "k" into k
(Places the value of the spring constant entered by the user into a variable)
if k = "" then
(Executed if the user fails to enter a spring constant, sets k to the default value of 1)
   put 1 into k
   put k into cd fld "k"
end if
put word 1 of cd fld "m" into m
(Places the value of the mass entered by the user into a variable)
if m="" then
(Executed if the user has failed to enter a mass value, sets m to the default value of 1)
   put 1 into m
   put m into cd fld "m"
end if
put word 1 of cd fld "tf" into tf
(Places the value of the time of execution into the time factor variable)
put (400-thumbPosition of me)/100 into disp
(Calculates the value of the displacement of the spring as input from the scroll bar)
set the numberFormat to "0.00"
(Sets the format in which numbers are displayed by the program)
put disp into cd field "Displacement"
(Places the initial displacement entered by the user on the screen)
plotGraph disp,k,m,tf
(Executes the main subroutine plotGraph and passes the four variables to it)
end mouseUp
```

The home button allows the user to exit the module. It resets critical variables and

repositions on screen graphics for future use. The code for the home button is shown below.

```
on mouseUp
  put "" into cd fld "k"
  (Empty the spring constant variable)
  put "" into cd fld "m"
  (Empty the mass variable)
  put 15 into cd fld "tf"
  (Place 15 into the time factor variable)
  put 0 into cd fld "Displacement"
  (Places a zero into the displacement card field)
  set the thumbPosition of scroll bar 1 to 400
  (Resets the scroll bar to the center position)
  set the loc of foreground draw object "mass" to loc of background   paint field "mass
  slider"
  (Moves the mass to the initial position)
  get the rect of paint field "graph plot"
  (Places the pixel location of the graphs into memory)
  choose select tool
  (Selects a graphics controlling tool)
  drag from item 1 of it, item 2 of it to item 3 of it item 4 of it
  (Places a selecting box around the current image)
  doMenu clear picture
  (Erases the current image from the screen)
  choose browse tool
  (Returns control to the user)
  run "diff.exe"
  (Starts the master program controlling structure)
  hide this window
  (Removes the window from the screen while it is being shut down)
  doMenu "save"
  (Saves the module)
  doMenu "quit"
  (Turns off the module)
end mouseUp
```

This ends the critical scripting for the undamped spring module.

LIST OF REFERENCES

1. Finney, Ross L., and Thomas, George B., *CALCULUS*, 8th Edition, Addison-Wesley Publishing Company, Reading, Massachusetts, 1992.

2. Sandefur, James T., *Discrete Dynamical Systems-Theory and Applications*, Oxford University Press, New York, New York, 1990.

3. *A User's Guide to WindowCraft*, Volume 1, Windowcraft Corporation, Acton, Massachusetts 1991.

4. Giordano, Frank R., and Weir, Maurice D., *A First Course in Mathematical Modeling*, Brooks/Cole Publishing Company, Monterey, California, 1985.

5. Goodman, Danny, *The Complete Hypercard Handbook*, 3rd Edition, Bantam Books, New York, New York, 1990.

6. Goodman, A. W., *Analytic Geometry and the Calculus*, Fourth Edition, MacMillan Publishing Compnay, Inc., New York, New York, 1980.

7. Appleton, Bill, *Supercard*, Silicon Beach, San Diego, California, 1993.

8. Reichard, Kevin, and Johnson, Eric F., *teach yourself... UNIX*, MIS:Press, New York, New York, 1992.

9. Cody, W.J., "Floating Point Parameters, Models and Standards," paper presented at the Argonne National Laboratory, Argonne, Illinois, 1982.

10. Boyce, William E. and DiPrima, Richard C., *Elementary Differential Equations and Boundary Value Problems*, 5th Edition, John Wiley & Sons, Inc., New York, New York, 1992.

11. Zill, Dennis G., *A First Course in Differential Equations with Applications*, 5th Edition, Prindle, Weber & Schmidt, Boston, Massachusetts, 1979.

12. Foley, J. D., and Van Dam, A., *Fundamentals of Computer Graphics*, Addison-Wesley Publishing Company, Reading, Massachusetts, 1982.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria VA 22304-6145	2
2.	Library, Code 052 Naval Postgraduate School Monterey CA 93943-5002	2
3.	Professor Maurice D. Weir, MA/Wc Department of Mathematics Naval Post Graduate School Monterey, CA 93943-5000	2
4.	Professor Carlos F. Borges, MA/Bc Department of Mathematics Naval Post Graduate School Monterey, CA 93943-5000	2
5.	Professor David Canright, MA/Ca Department of Mathematics Naval Post Graduate School Monterey, CA 93943-5000	2
6.	COL Frank Giordano Department of Mathematical Sciences United States Military Academy West Point, NY 10996-1786	1
7.	Professor Richard Franke, MA/Fe Chairman, Department of Mathematics Naval Post Graduate School Monterey, CA 93943-5000	1

8.   CPT Brian E. Pierson   4
       Department of Mathematical Sciences
       United States Military Academy
       West Point, NY 10996-1786

9.   CPT Jon L. Christensen   4
       Department of Mathematical Sciences
       United States Military Academy
       West Point, NY 10996-1786

10.   Professor J. Alan Adams   1
       Department of Mechanical Engineering
       United States Naval Academy
       Annapolis, MD 21402-5003